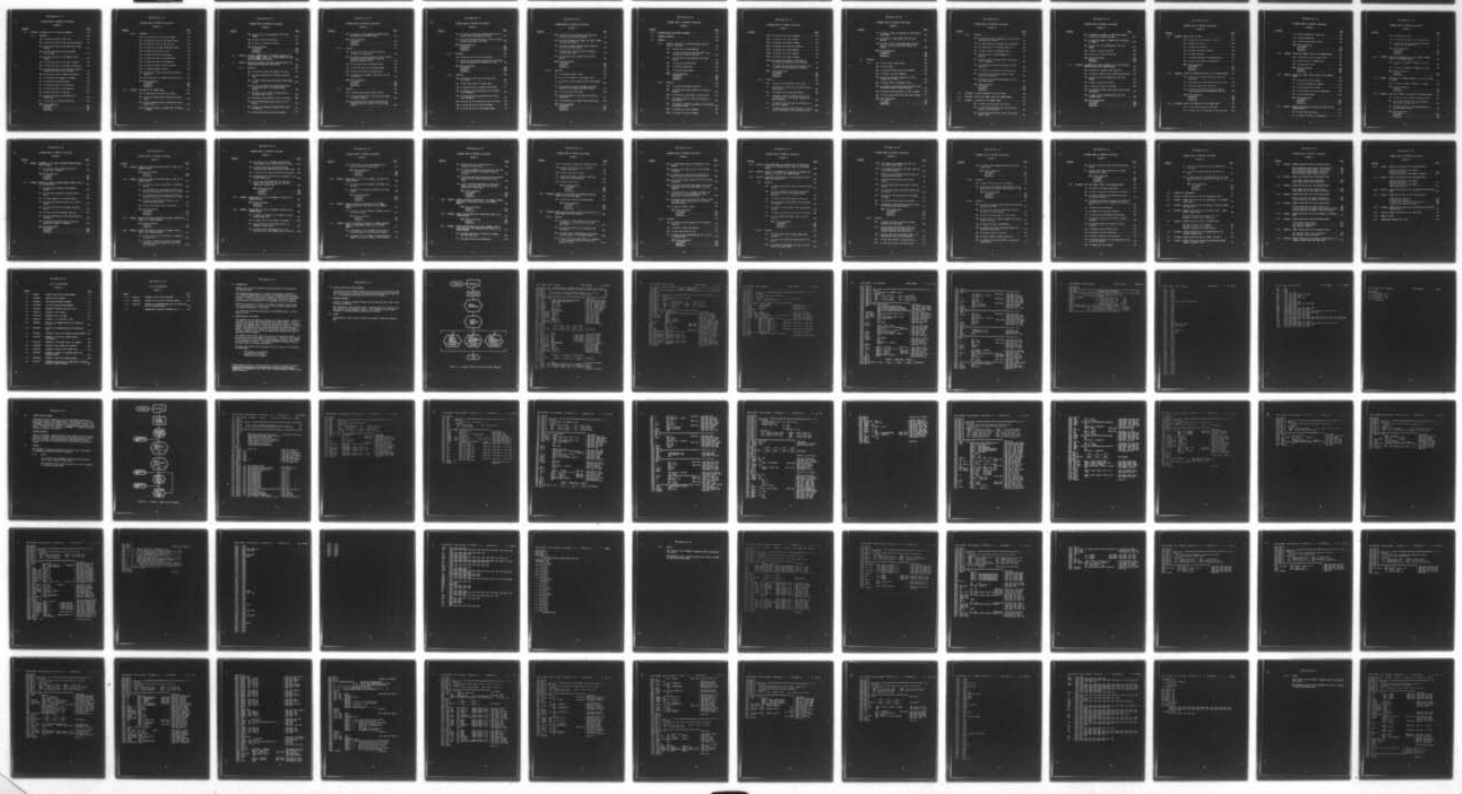
AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U).
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

1 OF 10
ADA
061811





AD A061611

TAEG

TRAINING ANALYSIS AND EVALUATION GROUP

8

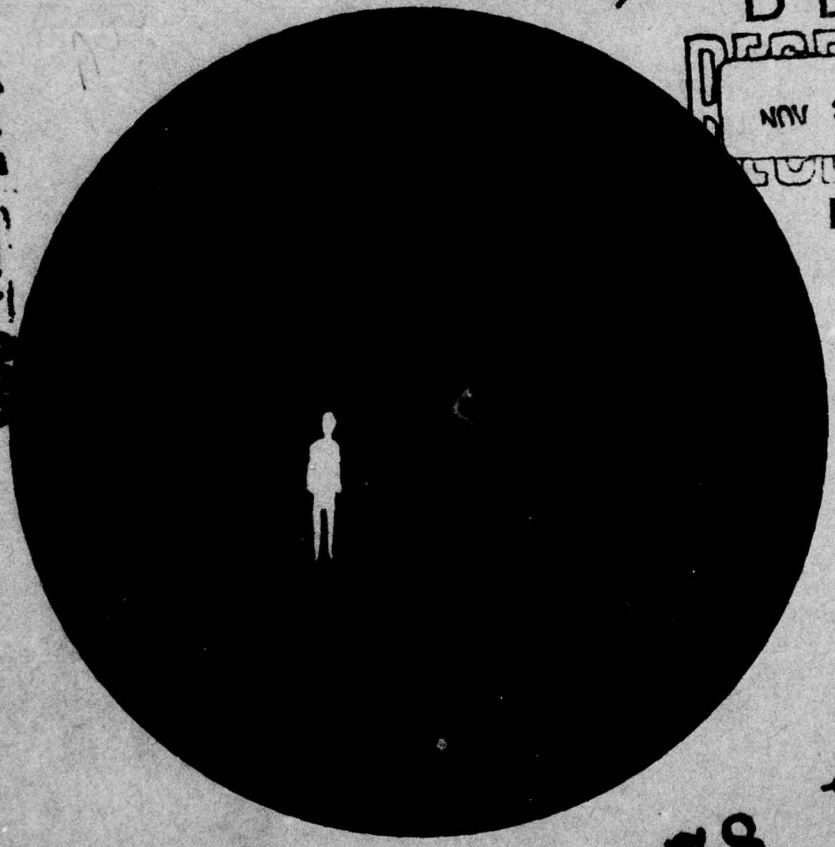
B. S.

TAEG REPORT NO. 59

COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE - VOLUME II

LEVEL II

DDC
NOV 28 1978
F



FOCUS ON THE TRAINED MAN

DDC FILE COPY

78 11 27 054

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION IS UNLIMITED.

JUNE 1978

TRAINING ANALYSIS AND EVALUATION GROUP
ORLANDO, FLORIDA 32813

TAEG Report No. 59

8

COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE, VOLUME II

Charles R. Guitard

AD A059572 Vol

TAEG-59-VOL-2

Training Analysis and Evaluation Group

June 1978

Final rept.,

884 p.

Sponsored by

Chief of Naval Education and Training

and the

David W. Taylor Naval Ship Research and Development Center,
Naval Technical Information Presentation Program

GOVERNMENT RIGHTS IN DATA STATEMENT

Reproduction of this publication in whole or in part is permitted for any purpose of the United States Government.

407 626

This document has been approved for public release and sale; its distribution is unlimited.

Alfred F. Smode

ALFRED F. SMODE, Ph.D., Director,
Training Analysis and Evaluation Group

Worth Scanland

WORTH SCANLAND, Ph.D.
Assistant Chief of Staff for
Research and Program Development
Chief of Naval Education and Training

407 626

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TAEG Report No. 59	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE, VOLUME II		5. TYPE OF REPORT & PERIOD COVERED Final Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Charles R. Guitard		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Training Analysis and Evaluation Group Orlando, FL 32813		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE June 1978 13. NUMBER OF PAGES 879
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Design of Instructional Materials
Instructional System Development
Computer Aided Authoring
Automated Authoring
Symbol Learning

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Publishing technical manuals and training handbooks for new Navy equipment is a high cost operation. In addition, current practices frequently result in the technical data being delivered late or in a form too complex for the reader to use. There are also problems in keeping technical data current and in distributing and storing the data where it is needed. → (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (continued)

The Naval Technical Information Presentation Program (NTIPP) is a 5-year multimillion dollar program to improve the Navy's efficiency in carrying out this publishing effort. This effort is managed by the David W. Taylor Naval Ship Research and Development Center with participation by other Navy activities. The Training Analysis and Evaluation Group (TAEG) represents the Chief of Naval Education and Training (CNET) in this program. The program is of special interest to CNET because the technical manuals serve as textbooks in the CNET managed "C" schools, and the training handbooks serve as the student and instructor guides in these schools. Improved documents will result in improved training.

This report contains an operator's handbook (Volume I) and program listing (Volume II) describing AUTHOR, a computer program for the automated authoring of programmed texts designed to teach symbol recognition.

Under NTIPP sponsorship, TAEG is pioneering an approach to authoring instructional materials that utilizes general purpose computers to carry out certain operations that are not being laboriously performed by hand. The goal is a technique for authoring effective training materials at a low cost. To demonstrate this approach, TAEG has written a program to aid in authoring materials for teaching symbol recognition. The human author enters information on a set of symbols into a data base. The computer routine merges this with general information on learning symbols and automatically organizes a complete learning package, including a pretest, discrimination drills, symbol recognition exercises, criterion tests, suggestions for refresher training, and an index. The output can be formatted for paper, microfiche, or video display. Using this procedure, authoring consumes a small portion of the time usually required for such projects. Instructional materials developed in this manner by TAEG are being evaluated in the Signalman "A" School and the onboard training program of the Naval Weather Service. The results to date are encouraging and will be formally published in subsequent TAEG reports. (--AD-A060 143).

An overview of the AUTHOR system is provided in a companion report (TAEG Report No. 58). Items included in that report are a discussion of the learning strategies used, the equipment required, and sample documents produced using AUTHOR.

TABLE OF CONTENTS

VOLUME II

		Page
1.0	INTRODUCTION.	39
2.0	AUTHOR SYSTEM INITIALIZING PROGRAM.	40
3.0	FORMAT DISPLAY PROGRAM.	51
4.0	DATA BASE MAINTENANCE PROGRAMS.	118
5.0	LEARNING MODULE MAINTENANCE PROGRAMS.	336
6.0	FORMAT DATA SUBPROGRAMS	782
7.0	FORMAT LISTINGS	832
8.0	COMPRESSING PROGRAM SOURCE CODE	878

ACCESSION for
NTIS White Section ☑
DDC Buff Section ☐
UNANNOUNCED ☐
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. SPECIAL
A

EXTENDED TABLE OF CONTENTS

VOLUME II

Section		Page
1.0	INTRODUCTION.	39
1.1	ORGANIZATION OF THE REPORT.	39
2.0	AUTHOR SYSTEM INITIALIZING PROGRAM.	40
2.1	PROGRAM FLOWCHART	40
2.2	AUTHOR-INITIALIZE DEFAULT PROGRAM DISK DRIVE.	40
	101 Cursor Address and Screen Erase.	43
	102 Load an Overlay Routine.	44
	104 Accept ALPHA and NUMERIC Input	45
	MAPS (Cross-reference)	
	Line Number.	48
	Variable	49
	Subroutine	50
3.0	FORMAT DISPLAY PROGRAM.	51
3.1	FORMAT FLOWCHART.	51
3.2	AUTHOR/1-CREATE AND MAINTAIN THE BASIC MODULE FORMAT DATA BASE	51
	3.2.1 ATH1/I	51
	Start	
	101 Cursor Address and Screen Erase	54
	102 To Load an Overlay Routine.	55
	104 Accept ALPHA and NUMERIC Input.	56
	108 To Output the Page Buffer onto the Selected Output Device	58
	109 To Scan the Text Buffer and Create a Lookup Table and Overlay Buffer for Selected Output Devices	60

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
	110	To Select Type of CRT Display to be Used. . . .	62
	111	To Display CRT Overlay Buffer onto Selected CRT Display	63
	112	To Display CRT Text Buffer onto Selected CRT Display	64
	113	To Display Instructions to Set Up TTY Devices	65
	MAPS	(Cross-reference)	
		Line Number	67
		Variable.	69
		Subroutine.	70
3.2.2	ATH2/I	. . . . .	71
	114	To Print Upper/Lower Hash Lines on the TTY Device.	72
	115	To Scan Lookup Table for Next Underscored Character	73
	116	To Print Line Containing Underscoring on TTY Device.	74
	116.1	To Increment and Compute the Index into the Underscore Lookup Table	76
	116.2	To Increment and Compute the Index into the "Z" Scan Lookup Table	77
	116.3	To Increment and Compute the Index into the "0" Scan Lookup Table	78
	117	To Print a Line Without Trailing Blanks on Selected Output Device.	79
	118	To Display Instructions to Set Up LPT Device. .	80

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
	119	To Print Upper/Lower Hash Lines on the LPT Device.	83
	120	To Print the Underscore Overlay Buffer on the LPT Device.	84
	121	To Clear All Tab Stops and Set the Needed Tab Stops	85
	122	To Allow the Output Device to be Reset.	86
	123	To Load 32 Line Text Buffer for 16 Line Input Buffer.	87
	MAPS	(Cross-reference)	
		Line Number	88
		Variable.	89
		Subroutine.	90
3.2.3	ATH3/I.		91
	124	To Select the Type of Special Character Scanning to be Executed.	92
	125	To Scan Search Return Buffer and Insert Special Character into the Buffer	93
	126	To Print the Text String Buffer X9$( )	94
	MAPS	(Cross-reference)	
		Line Number.	95
		Variable	96
		Subroutine	97
3.2.4	ATH4/I.		98
	190	A General Nine-Track Utility Package	99
	191	To Mask the GIO Register Buffer with Error Mask-Buffer and Display Error Messages, if any.	101
	MAPS	(Cross-reference)	
		Line Number.	103
		Variable	104
		Subroutine	105

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
	3.2.5	ATH5/I.	106
		200 To Initialize Program Tables	107
		201 To Initialize Program Input and Output Devices .	109
		202 To Load Format File.	111
		205 To Load and Display Data Subprogram on Currently Selected Output Device	112
		206 To Read the Data Subprogram and Load into Storage Buffer	113
		MAPS (Cross-reference)	
		Line Number.	115
		Variable	116
		Subprogram	117
4.0	DATA BASE MAINTENANCE PROGRAMS.		118
4.1	PROGRAM FLOWCHART		118
4.2	AUTHOR/2-CREATE AND MAINTAIN THE AUTHOR PHASE II DATA BASE. . .		129
	4.2.1	ATH1/II	129
		Start	130
		000 Error Recovery	132
		101 Cursor Address and Screen Erase.	133
		102 To Load and Overlay Routine.	134
		104 To Accept ALPHA and NUMERIC Input.	135
		105 To Display Program Directives and Accept Input .	137
		MAPS (Cross-reference).	138
		Variable	139
		Subroutine	141

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
4.2.2	ATH2/II.		142
	106	To Accept and Store Multiple Lines of Data . .	143
	107	To Initialize Pointers and Buffer.	144
	108	To Input a Character and Verify for Validity .	145
	109	To Insert Character into Buffer and Process any Control Functions.	146
	110	To Transfer Character from Character Buffer to Storage Buffer.	147
	111	To Rubout Current Character.	149
	112	To Underscore the Current Character.	150
	113	To Allow the Storage Buffer to be Edited . . .	151
	114	To Position Cursor Left or Right of Current Position	153
	115	To Position Cursor Up or Down of Current Position	154
	116	To Change Current Character Only	155
	117	To Delete a String of Characters	156
	118	To Insert Characters into the Buffer	157
	119	To Position the Cursor at the Beginning or End of the Text Buffer	158
	120	To Compute the Cycle Number.	159
	121	To Roll Screen When Position is Advanced Off the Screen	160
	122	To Load Storage Buffer with Text Before Editing.	161
	123	To Set End of Data Flag and Save Buffer on Disk	162

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		MAPS (Cross-reference)	
		Line Number	163
		Variable	165
		Subroutine	166
	4.2.3	ATH3/II	167
		124 To Position Cursor and Buffer Pointer to Next Line	168
		125 To Load Storage Buffer from Disk Given Beginning Sector Address	169
		129 To Write Storage Buffer to Disc Updating Disk Index	170
		130 To Search Disk Index for Item ID	171
		131 To Erase Existing Data from Disk	172
		132 To Copy Data from Storage Buffer to Disk	173
		133 To Set Up for a Disk Transfer so as to Link Data to the End of Existing Data on Disk	174
		134 To Update Disk Index and ID Memory Table with Current Item ID and Beginning Disk Location	175
		135 To Reset Address Registers to the Next Available Sector Address	176
		136 To Locate the Last Character in the Buffer and Return the Total Length of the Buffer in Variable R0	177
		MAPS (Cross-reference)	
		Line Number	178
		Variable	179
		Subroutine	180
4.3	ATH2INIT	INITIALIZE THE PROGRAM VARIABLES AND DATA BASE DISKETTE	181
	4.3.1	ATH2INT1	181

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section		Page
	200 To Initialize Program Tables.	183
	201 To Display Title Page on Console Device	186
	202 To Display the Command Summary and Accept Input Command	187
	203 To Process the Input Command.	188
	MAPS (Cross-reference)	
	Line Number	189
	Variable.	190
	Subroutine	191
4.3.2	ATH2INT2	192
	204 To Initialize Disk Index for Initial Entry of Data Base	193
	205 To Verify that the Proper Data Base Disk is Mounted	195
	206 To Create a Backup Copy of the AUTHOR Data Base .	196
	207 To Assign an Input Disk Device to Device Table. .	198
	208 To Wait for Operator Response Displaying Elapsed Time of Wait.	199
	209 To Provide the Basic Sequence for Initializing the Data Entry Process.	200
	210 To Input the Device Structure to be Assigned. . .	201
	211 To Assign Device to Internal Table Slot	203
	MAPS (Cross-reference)	
	Line Number	204
	Variable.	206
	Subroutine.	207

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
4.4	ATH2LOGC	Provides Logic for Executing Commands.	208
	4.4.1	ATH2LOG1	208
		201 To Allow the Entry of the Title	210
		202 To Allow the Entry of the Training Objective. .	211
		203 To Allow the Entry of the Importance of the Symbols	212
		204 To Allow the Entry of How the Symbols Will be Used.	213
		205 To Allow the Entry of an Example of the Symbols	214
		206 To Allow the Entry of the Type of Symbol. . . .	215
		207 To Allow the Entry of the Name of the Users . .	216
		208 To Allow the Entry of the Official Source of Reference for Definition of Symbols	217
		209 To Allow the Entry of Symbol Descriptions . . .	218
		211 To Re-display the Summary of Commands	219
		212 To Terminate Program Execution.	220
		213 To Allow the Entry of the Subtitles	221
		214 To Allow the Entry of the Author(s)	222
		215 To Allow the Entry of the Date.	223
		216 To Allow the Entry of the Name of the Organization.	224
		217 To Load the List Processing Routine	225
		MAPS (Cross-reference)	
		Line Number	226
		Variable.	227
		Subroutine.	228

EXTENDED TABLE OF CONTENTS (continued)

Volume II

Section			Page
	4.4.2	ATH2LOG2.	230
		218 To Allow the Entry of the Set Number	231
		219 To Allow the Entry of the Symbol Number. . . .	232
		220 To Allow the Entry of the Set Name	233
		221 To Construct the Symbol Reference Number . . .	234
		222 To Store Entered Data on Disk.	235
		223 To Allow the Entry of the Symbol	236
		224 To Allow the Entry of the Definition	237
		225 To Allow the Entry of the Mnemonic	238
		226 To Allow the Entry of Similar Symbols	239
		227 To Increment Symbol Counter.	240
		228 To Delete All Items for Currently Selected Symbol ID.	241
		230 To Display Menu for Symbol Description Input .	242
		MAPS (Cross-reference)	
		Line Number.	243
		Variable	244
		Subroutine	245
4.5	ATH2LIST	INITIALIZES THE SYMBOL TABLE.	246
		201 To Define and Initialize Sort Tables	248
		202 To Load the Merge Buffer from the Data Base Index.	249
		203 To Sort the Merge Buffer Creating the Sorted Symbol Table	250
		206 To Count the Number of Symbol Sets and Number of Symbols	251

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		207 To Set a Bit on Corresponding to the Item Number.	252
		208 To Mask a Given Bit Position.	253
		209 To Turn on a Given Bit Position	254
		MAPS (Cross-reference)	
		Line Number	255
		Variable.	256
		Subroutine.	257
4.6	ATH2LSTV	DEFINES A COMMON TABLE FOR VARIABLES NEEDED IN THE LISTING PROGRAM 'COMTBL'--TO CLEAR PROGRAM TEXT AND LOAD NEW COMMON DEFINITIONS.	258
4.7	ATH2LSTP	PROVIDE THE CONTROL LOGIC AND LISTING ROUTINES FOR OUTPUTTING DATA BASE ON LINE PRINTER.	260
	4.7.1	ATH2LST1	260
		106 To Load the Printer Page Buffer with Text . . .	263
		107 To Load Storage Buffer with Data Item from Disk.	264
		108 To Output Page Buffer onto the Selected Output Device.	265
		109 To Scan Text Buffer and Create Lookup Table and Overlay Buffer for Selected Output Devices	266
		117 To Print a Line Without Trailing Blanks on Selected Output Device.	268
		118 To Display Instructions to Set Up LPT Device. .	269
		119 To Print Upper/Lower Hash Lines on the LPT Device.	270
		120 To Print the Underscore Overlay Buffer onto LPT Device.	271
		121 To Allow the Output Device to be Reset.	272

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section		Page
	123 To Develop a Large CURSOR on Console Screen While Waiting for Device Ready.	273
	MAPS (Cross-reference)	
	Line Number	274
	Variable.	
	Subroutine.	276
4.7.2	ATH2LST2	277
	124 To Select the Type of Special Character Scanning to be Executed	278
	125 To Scan Search Return Buffer and Insert Special Character into Text Buffer in Location Specified by Buffer Value	279
	126 To Print the Text String Buffer X9$( ).	280
	127 To Test the Tape Unit for Ready Status.	281
	128 To Test the Printer for Ready Status.	282
	129 To Display the Alignment Instructions on the Line Printer.	283
	MAPS (Cross-reference)	
	Line Number	288
	Variable.	289
	Subroutine.	290
4.7.3	ATH2LST3	291
	130 To Search Disk Index for Data Item ID	292
	131 To Load Storage Buffer from Disk Given Beginning Sector Address	293
	132 To Create a Block of Text from Data in the Storage Buffer and Transfer it to the Page Buffer.	294

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section		Page
	133 To Format the Input Word Within Margins and to Control Page Formatting and Storage.	294
	134 To Load the Page Buffer with a Title and Format the Data Base Item on the Page	296
	MAPS (Cross-reference)	
	Line Number.	297
	Variable	298
	Subroutine	299
4.7.4	ATH2LST4.	300
	190 A General Nine-Track Utility Package	301
	191 To Mask the GIO Register Buffer with Error Mask Buffer and Display Error Messages, if any. . . .	303
	MAPS (Cross-reference)	
	Line Number.	305
	Variable	306
	Subroutine	307
4.7.5	ATH2LST5.	308
	200 To Initialize the Line Printer and Print Title Page	309
	201 To Test the Printer for Ready Status	310
	202 To Allow the Entry of Symbol Descriptions. . . .	311
	203 To Display a Brief Description of the Phase 2 List Procedures	312
	204 To Allow the Entire Data Base to be Listed (LIST Commands).	313
	205 To Print the Final Page onto the Line Printer .	314
	206 To Allow the Entry of the Set Number	315
	207 To Allow the Entry of the Symbol Number.	316

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section		Page
	208 To Print the Current Symbol Set Name After Verifying Set and Symbol Number	317
	209 To Load the Symbol Set Number and Symbol Number into the Page Buffer.	318
	210 To Load the Symbol and the Symbol Reference Code into the Page Buffer	319
	211 To Load Either the Definition or Mnemonic into the Page Buffer.	320
	212 To Load the Similar Symbol into the Page Buffer.	321
	MAPS (Cross-reference)	
	Line Number	322
	Variable.	323
	Subroutine.	324
4.7.6	ATH2LST6	325
	213 To Increment Symbol Counter	326
	214 To List All Symbols in the Symbol Table	327
	215 To Perform a Disk Lookup of the Current Symbol ID.	328
	216 To Compute the Symbol Set Number and Symbol Number from the System Reference Number	329
	217 To Test Current Line Index.	330
	218 To Load the Name of the Symbol Set into the Page Buffer	331
	219 To Display Menu for Symbol Description Input.	332
	MAPS (Cross-reference)	
	Line Number	333
	Variable.	334
	Subroutine.	335

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
5.0	LEARNING MODULE MAINTENANCE PROGRAMS.		336
5.1	PROGRAM FLOWCHARTS.		336
5.2	AUTHOR/3.		336
5.2.1	ATH1/III Construct a Learning Package from the Phase II Data Base.		336
	091	To Wait for an Input Response.	352
	100	To Load a String into the Page Buffer and Display the String on the CRT.	353
	101	To Provide for Cursor Address and Screen Erase.	354
	102	To Load an Overlay Routine	355
	103	To Perform Conversions	356
	104	To Perform Index Lookup.	357
	MAPS	(Cross-reference)	
		Line Number.	358
		Variable	359
		Subroutine	361
5.2.2	ATH2/III.		362
	105	To Load Storage Buffer from Disk	363
	106	To Verify that Proper Output Disk Volume is Mounted.	364
	106.1	To Initialize a New Phase III Output Disk. . .	365
	107	To Create a Block of Text from Storage to Page Buffer.	367
	108	To Execute a Formatting Command by Positioning the Buffer Pointer	368
	108.1	To Execute the < text > Command.	369
	108.2	To Execute the < word > Command.	370

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
	108.3	To Execute the <lower> Command.	371
	108.4	To Execute the <caps> Command	372
	108.5	To Execute the <table> Command.	373
	108.6	To Execute the <subtitle> Command	374
	108.7	To Execute the <page> Command	375
	108.8	To Execute the <line> Command	376
	108.9	To Execute the <space> Command.	377
	108.10	To Compute the Length of the Buffer to Which Modifications are to be Performed . . .	378
	109	To Format the Input Word Within Margins and to Control Page Formatting and Storage. . . .	379
	MAPS	(Cross-reference)	
		Line Number	380
		Variable.	381
		Subroutine.	382
5.2.3	ATH3/III		383
	110	To Output the Page Buffer to the Disk	384
	111	To Locate Next Index Slot and Load Previous Page Address.	385
	112	To Compute the Current Page Address	386
	113	To Reserve Space for Index at End of Each Filled Fiche if a Fiche Index Sequence Has Been Requested.	387
	114	To Search the Output Page for References to the 'NXT' Command	388
	115	To Update Index and Index Pointers; To Insert Current Page Address and Save Page on Disk. .	389

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section				Page
		117	To Search a Page for Reference to a Redefined Page Number	390
		118	To Construct a Page Number from the Page Address	391
		119	To Check if There are Enough Pages Left on a Row of a Fiche to Insert the Ordered Page Formats	392
		MAPS	(Cross-reference)	
			Line Number	393
			Variable	394
			Subroutine	395
5.3	ATH3INIT			396
	5.3.1	201	To Initialize Program Tables	398
		202	To Print Header	401
		203	To Display Program Options and Commands	402
		204	To Process the Input Commands	403
		205	To Wait for Operator Response Displaying Elapsed Time of Wait	404
		206	To Accept ALPHA and NUMERIC Inputs	405
		207	To Assign an Input Disk Device to Device Table and Verify Disk Has Proper Contents	407
		208	To Display Formatting Menu and Input Command	408
		209	To Display Operational Menu and Input Command	411
		MAPS	(Cross-reference)	
			Line Number	412
			Variable	414
			Subroutine	415

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
5.3.2	ATH3INT2.		416
	210	To Provide the Basic Sequence for Initializing the Data Entry Process	417
	212	To Input the Device Structure to be Assigned .	418
	213	To Assign Device to Internal Table Slot. . . .	420
	214	To Verify Contents of Mounted Disks.	421
	215	To Verify that a Standard WANG Disk is Mounted.	422
	216	To Verify that the Proper Phase 2 Data Base Disk is Mounted.	423
	217	To Initialize and Verify the Phase III Data Base Disk.	424
	218	To Initialize the Phase III Data Base Disk . .	425
	219	To Verify that the Proper Phase III Data Base Disk is Mounted	427
	220	To Assign the Starting Frame Number for the Learning Package	428
	MAPS	(Cross-reference)	
		Line Number.	429
		Variable	431
		Subroutine	432
5.4	ATH3STAB	TO CLEAR FROM MEMORY TEXT NOT NEEDED.	433
5.5	ATH3VTAB	TO LOAD THE SYMBOL TABLE INTO COMMON MEMORY	435
5.6	ATH3SRTS	TO INITIALIZE THE SYMBOL TABLE.	437
	201	To Define and Initialize Sort Tables	439
	202	To Load the Merge Buffer from the Data Base Index; To Sort Each Row.	440
	203	To Sort the Merge Buffer Creating the Sorted Symbol Table	441

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		205 To Compute the Number of Index Pages Needed at the End of Each Fiche Page.	442
		206 To Count the Number of Symbol Sets and Numbers of Symbols	443
		207 To Set a Bit on Corresponding to the Item Number	444
		208 To Mask a Given Bit Position	445
		209 To Turn on a Given Bit Position.	446
		MAPS (Cross-reference).	447
		Variable	448
		Subroutine	449
5.7	ATH3AUTO	TO PROVIDE THE OVERLAY CONTROL LOGIC FOR THE MAJOR CONSTRUCTION OF THE LEARNING MODULE	450
		200 To Generate a Random Linear Subscript.	452
		201 To Locate a Subscript Not Previously Generated .	453
		202 To Set a Bit on Corresponding to the Item Number	454
		203 To Mask a Given Bit Position	455
		204 To Turn on a Given Bit Position.	456
		205 To Initialize Output Page Buffer from the Data Subprogram	457
		206 To Read the Data Subprogram and Load into Storage Buffer	458
		MAPS (Cross-reference)	
		Line Number.	460
		Variable	461
		Subroutine	462

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
5.8	ATH3ØØ.Ø	CREATE THE TITLE PAGE.	463
		210 To Save Text to be Stored on Line 16.	465
		211 To Create the Title	466
		212 To Create the Subtitle.	467
		213 To Create the Author Credits.	468
		214 To Create the Date.	469
		215 To Create the Name of the Organization.	470
		216 To Save Current Page on Output Disk	471
		MAPS (Cross-reference)	
		Line Number	472
		Variable.	473
		Subroutine.	474
5.9	ATH3Ø1.Ø	CREATE THE INTRODUCTION PAGE OF THE LEARNING MODULE.	475
		210 To Save Text on Lower Portion of Page Then Clear Page.	477
		211 To Load the Training Objective onto the Page. .	478
		212 To Re-load Page with Fixed Text	479
		213 To Save Current Page and Clear in Order to Allow the Loading of the Remainder of Text. . .	480
		MAPS (Cross-reference)	
		Line Number	481
		Variable.	482
		Subroutine.	483
5.10	ATH3Ø2.Ø	CREATE THE IMPORTANCE OF THE SYMBOL PAGE	484
		210 To Save Lower Heading of the Page	486
		211 To Write Text on Importance of Learning Task. .	487

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		212 To Write Heading for Usage Text.	488
		213 To Write Usage Text.	489
		214 To Save Current Page on Disk	490
		MAPS (Cross-reference)	
		Line Number.	491
		Variable	492
		Subroutine	493
5.11	ATH3Ø3.Ø	CREATE THE EXAMPLE PAGE OF THE LEARNING MODULE. . .	494
		210 To Save Text on Last Line of Page.	496
		211 To Write Text for Example Page	497
		212 To Save Page on Disk	498
		MAPS (Cross-reference).	499
		Variable	500
		Subroutine	501
5.12	ATH3Ø4.Ø	CREATE THE SYMBOL PRETEST PAGE OF THE LEARNING MODULE.	502
		210 To Write the Type of Symbol.	504
		211 To Write a Sampling of Symbols	505
		212 To Save Lower Part of Page	506
		213 To Write Official Source of Reference.	507
		214 To Save Current Page	508
		MAPS (Cross-reference)	
		Line Number.	509
		Variable	510
		Subroutine	511
5.13	ATH3Ø4.1	CREATE THE OPTIONAL CRITERION TEST PAGES OF THE LEARNING MODULE	512
		210 To Save Lower Page Text.	514
		211 To Create Criterion Test Questions	515

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		212 To Save Page on Disk.	515
		213 To Create the Answers for Criterion Test. . . .	517
		214 To Calculate the Lowest Passing Score and Insert into the Page Text	518
		MAPS (Cross-reference)	
		Line Number.	519
		Variable	520
		Subroutine	521
5.14	ATH3Ø5.Ø	CREATE THE ORGANIZATION OF THE LEARNING PROGRAM PAGE OF THE LEARNING MODULE	522
		210 To Insert the Number of Symbol Sets into Text	524
		211 To Insert the Number of Symbols into Text. . .	525
		MAPS (Cross-reference).	
		Variable	526
		Subroutine	527
5.15	ATH3Ø6	INITIALIZE THE PROGRAMS NEEDED TO CREATE THE SYMBOL SET PAGES	528
		210 To Load the Symbol Set Number in Heading	530
		MAPS (Cross-reference)	
		Variable	531
		Subroutine	532
5.16	ATH3Ø6.Ø	CREATE THE SYMBOL LIST PAGE OF THE LEARNING MODULE	533
		211 To Save the Text in Lower Portion of Page. . .	535
		212 To Load Text Buffer with List of Symbols Contained in Current Set	536
		213 To Load Fixed Text in the Remaining Space and Save Page on Disk.	537
		MAPS (Cross-reference)	
		Line Number.	538
		Variable	539
		Subroutine	540

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
5.17	ATH3Ø6.1	DETERMINE IF THE "NOTE DIFFERENCES AMONG SYMBOLS" PAGE IS NEEDED.	541
		211 To Find Any Similar Symbols Within the Current Set of Symbols	543
		MAPS (Cross-reference)	
		Line Number.	544
		Variable	545
		Subroutine	546
5.18	ATH3Ø6.2	CREATE THE "NOTE DIFFERENCES AMONG SYMBOLS" PAGE OF THE LEARNING MODULE	547
		211 To Load Similar Symbols into Temporary Storage.	549
		212 To Save Text Contained in Lower Portion of Page	550
		213 To Insert Symbol List into Page Buffer	551
		214 To Save Page Containing the List of Similar Symbols.	552
		215 To Insert Current Similar Symbol into Page Buffer	553
		216 To Create the Discrimination Exercises	554
		217 To Create the Answers to the Discrimination Exercises.	556
		218 To Minimize the Chance that the Same Two Symbols Will Appear in Sequence.	557
		MAPS (Cross-reference)	
		Line Number.	558
		Variable	559
		Subroutine	560

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
5.19	ATH3Ø6.3	CREATE THE SYMBOL DEFINITIONS (PART 1) PAGE OF THE LEARNING MODULE	561
		MAPS (Cross-reference)	
		Variable	563
		Subroutine	564
5.20	ATH3Ø6.4	CREATE THE SYMBOL DEFINITIONS (PART 2) PAGE OF THE LEARNING MODULE	565
		211 To Save Last Line of Text Format in Temporary Storage.	567
		212 To Load Memory Aid into Temporary Buffer and Determine Approximate Size of Text (in lines).	568
		213 To Load Definition into Temporary Buffer and Determine Approximate Size of Text (in lines).	569
		214 To Insert the Definition, Memory Aid, and Symbol into Page Buffer.	570
		215 To Save Final Page of Text on the Disk	571
		MAPS (Cross-reference)	
		Line Number.	572
		Variable	573
		Subroutine	574
5.21	ATH3Ø6.5	CREATE THE PRACTICE DEFINING THE SYMBOL INTRODUCTION PAGE OF THE LEARNING MODULE	575
		MAPS (Cross-reference)	
		Variable	577
		Subroutine	578
5.22	ATH3Ø6.6	CREATE THE PRACTICE DEFINING THE SYMBOLS EXERCISE PAGE OF THE LEARNING MODULE	579
		211 To Save Last Line of Text Buffer in Temporary Storage Buffer	581
		212 To Insert a Selection of Symbols and Symbol Definitions from the Symbol Set Table into the Page Buffer.	582

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		213 To Create a List of Symbols Selected from Those Symbols Used in Lower Portion of Frame. . .	583
		214 To Save Current Page and Insert Message Indicating that Section Continues on Next Page. .	584
		215 To Save the Last Page of the Practice Sequence. .	585
		216 To Generate and Store the List of Symbols in the Current Symbol Set	586
		217 To Fill the Space Remaining on a Page with a Symbol from Current Set that has not been Inserted on Current Page	588
		MAPS (Cross-reference)	
		Line Number	588
		Variable.	589
		Subroutine.	590
5.23	ATH3Ø6.7	CREATE PART 1 OF THE TEST YOURSELF SECTION OF THE LEARNING MODULE	591
		MAPS (Cross-reference)	
		Variable	593
		Subroutine	594
5.24	ATH3Ø6.8	CREATE PART 2 OF THE TEST YOURSELF SECTION OF THE LEARNING MODULE	595
		211 To Generate a Random List of Symbols from the Current Symbol Set	597
		212 To Insert the List of Random Test Symbols. . . .	598
		213 To Save the Last Line of Page Buffer in a Temporary Storage Buffer	599
		214 To Insert Either the Memory Aids or the Definitions for Those Symbols Used in the Test .	600

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		215 To Save Page on Disk After Reloading Last Line of Text from Storage Buffer.	601
		MAPS (Cross-reference)	
		Line Number.	602
		Variable	603
		Subroutine	604
5.25	ATH3Ø6.9	CREATE PART 3 OF THE TEST YOURSELF SECTION OF THE LEARNING MODULE.	605
		211 To Insert the Current Symbol Set Number into the Text.	607
		212 To Check if Last Section Symbol Set Has Been Created	608
		MAPS (Cross-reference)	
		Line Number.	609
		Variable	610
		Subroutine	611
5.26	ATH3Ø7.0	CREATE THE PRACTICE DEFINING ALL THE SYMBOLS (OVERVIEW AND DIRECTIONS) PAGE OF THE LEARNING MODULE	612
		210 To Insert the Total Number of Symbols into the Text of the Format.	614
		MAPS (Cross-reference)	
		Variable	615
		Subroutine	616
5.27	ATH3Ø7.1	CREATE THE PRACTICE DEFINING ALL SYMBOLS (PRACTICE EXERCISES AND ANSWERS) PAGES OF THE LEARNING MODULE	617
		210 To Generate a List of Symbols to be Used as Practice Symbols in the Two Frame Sequence. . . .	619
		211 To Create a List of Symbols Selected from the Symbols to be Used on Page 2 of the Sequence. . .	620

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		212 To Save Last Line of Text Buffer in a Temporary Storage Buffer.	621
		213 To Insert the Answers to the Practice Exercises in a Sorted Manner on the Second Frame of Sequence.	622
		214 To Save Current Page on Disk and Insert Message Indicating that Section Continues on Next Page.	623
		215 To Fill the Space Remaining on a Page with a Symbol from Current Set that Has Not Been Inserted on Current Page.	624
		MAPS (Cross-reference)	
		Line Number.	625
		Variable	626
		Subroutine	627
5.28	ATH307.2	CREATE THE PRACTICE DEFINING ALL THE SYMBOLS (ENDING AND PRACTICE DIRECTIONS) PAGE OF THE LEARNING MODULE	628
		MAPS (Cross-reference)	
		Variable	630
		Subroutine	631
5.29	ATH308.0	CREATE THE CRITERION TEST (DIRECTIONS) PAGE OF THE LEARNING MODULE.	632
		MAPS (Cross-reference).	634
		Variable	635
		Subroutine	636
5.30	ATH308.1	CREATE THE CRITERION TEST (TESTS, ANSWERS, END OF TEST DIRECTIONS, AND CONGRATULATIONS) PAGES OF THE LEARNING MODULE.	637
		210 To Insert the Current Criterion Test Number into the Frame Text	639
		211 To Create Criterion Test Questions.	640

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		212 To Create the Answers for Criterion Tests. . .	641
		213 To Erase and Position Text to Reflect Proper Instructions.	642
		214 To Save Lower Part of Page	643
		215 To Write the Name or Rating of the Group Using This Learning Module	644
		216 To Save Current Page	645
		MAPS (Cross-reference)	
		Line Number	646
		Variable.	647
		Subroutine.	648
5.31	ATH3Ø9.Ø	CREATE THE SUGGESTIONS FOR REFRESHER TRAINING PAGES OF THE LEARNING MODULE.	649
		210 To Save Last Frame of Learning Module.	651
		MAPS (Cross-reference)	
		Line Number	652
		Variable.	653
		Subroutine.	654
5.32	ATH3INDX	UPDATE THE DISK LOOKUP INDEX ON EACH DISKETTE IF A MULTIVOLUME OUTPUT WAS CREATED.	655
	5.32.1	ATH3IDX1	
		201 To Update All Index Sectors on All Volumes of the Output File if More than One Volume Was Created.	657
		202 To Load Page Buffer with First/Next Frame From Disk.	659
		203 To Check Current Page Address and to Create the Index Pages if Fiche Format Output	660
		204 To Scan the Current Page Buffer for a Keyword; To Set Up Keyword Buffer; To Process All Keywords on Page.	661

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section		Page
	205 To Insert Name Address Corresponding to the Keyword	662
	206 To Save Current Page on Disk if Any Changes Were Made	663
	208 To Update Disk Index and Index Pointers for Spaces Occupied by the Fiche Index Frames	664
	209 To Initialize Output Page Buffer from the Data Subprogram	665
	210 To Convert the Binary Page Address into Proper Format and Insert into Page Buffer at Given Location	666
	211 To Create the Lists of Symbols by Symbol Set to be Used in the Fiche Index and Insert These Lists into the Page Buffer	667
	212 To Insert an Index Item into the Index if There is Enough Space, Otherwise Skip to the Next Page Then Insert	669
	213 To Save Last Page of Index	670
	MAPS (Cross-reference)	
	Line Number	671
	Variable	672
	Subroutine	674
5.32.2	ATH3IDX2	675
	214 To Perform the Skip Forward or Skip Backward Function	676
	215 To Assign a Binary Page Address	677
	216 To Save Index Frame on Disk	678
	217 To Read the Data Subprogram and Load it into a Storage Buffer	679
	MAPS (Cross-reference)	
	Line Number	680
	Variable	681
	Subroutine	682

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
5.33	ATH3LIST	CLEAR FROM MEMORY ALL PROGRAM TEXT NOT NEEDED FOR LISTING THE LEARNING MODULE; LOAD IN A NEW COMMON TABLE.	683
5.34	ATH3LSTV	DEFINE A NEW COMMON TABLE AND LOAD THE PROGRAM TEXT NEEDED TO LIST/EDIT THE LEARNING MODULE.	685
5.35	ATH3LSTP	ALLOW THE LEARNING MODULE TO BE DISPLAYED ON ONE OF THREE OUTPUT DEVICES	687
	5.35.1	ATH3LST1	687
		108 To Output Page Buffer onto the Selected Output Device.	689
		109 To Scan Text Buffer and Create Lookup Table and Overlay Buffer for Selected Output Devices	691
		110 To Select Type of CRT Display to be Used. . . .	693
		111 To Display CRT Overlay Buffer onto Selected . . CRT Display	644
		112 To Display CRT Text Buffer onto Selected CRT Display	695
		113 To Display Instructions to Set Up TTY Device. .	696
		114 To Print Upper/Lower Hash Lines on the TTY Device.	698
		MAPS (Cross-reference) Line Number	699
		Variable	700
		Subroutine.	701
	5.35.2	ATH3LST2	702
		115 To Scan Lookup Table for Next Underscored Character	703
		116 To Print Line Containing Underscoring on TTY Device.	704
		116.1 To Increment and Compute the Index into the Underscore Lookup Table	706

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
	116.2	To Increment and Compute the Index into the Z Scan Lookup Table	707
	116.3	To Increment and Compute the Index into the Ø Scan Lookup Table	708
	117	To Print a Line Without Trailing Blanks on Selected Output Devices	709
	118	To Display Instructions to Set Up LPT Device.	710
	119	To Print Upper/Lower Hash Lines on the LPT Device.	111
	120	To Print the Underscore Overlay Buffer onto LPT Device.	712
	121	To Clear All Tab Stops and Set the Needed Tab Stops	713
	122	To Allow the Output Device to be Reset.	714
	123	To Display a Large Cursor on Console Screen While Waiting for Device Ready.	715
		MAPS (Cross-reference)	
		Line Number	716
		Variable.	717
		Subroutine.	718
5.35.3		ATH3LST3	719
	124	To Select the Type of Special Character Scanning to be Executed	720
	125	To Scan Search Return Buffer and Insert Special Character into Text Buffer in Location Specified by Buffer Value.	721
	126	To Print the Text String Buffer X9$( ) as Specified by Input String Parameters.	722
	127	To Test the Tape Unit for Ready Status.	723
	128	To Test the Printers for Ready Status	724

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section		Page
	129 To Display the Alignment Instructions on the Line Printer	725
	MAPS (Cross-reference)	
	Line Number	730
	Variable	731
	Subroutine	732
3.35.4	ATH3LST4	733
	190 A General Nine-Track Utility Package	734
	191 To Mask the GIO Register Buffer with the Error Mask Buffer and Display Error Messages, if any.	736
	MAPS (Cross-reference)	
	Line Number	738
	Variable	739
	Subroutine	740
5.35.5	ATH3LST5	741
	201 To Select the Output Device onto Which the Data Base Will be Output	742
	202 To Initialize Text and Index Buffers Before Beginning List Processing	743
	203 To Load First/Next Page into Text Buffer	745
	204 To Scan Buffer for Graphic Keyword; To Process Graphic Function	747
	205 To Compute the Next Frame Address	748
	206 To Perform and Index Lookup and Compute the Location of Next Page	749
	207 To Load Text Buffer from Disk	750
	208 To Process Special Graphic Functions	751
	209 To Compute Binary Page Address from Physical Page Address	752

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		210 To Print Text Buffer Onto the Selected Output Device	753
		211 To Check That Right Output Disk is Mounted Before Sector Lookup	754
		MAPS (Cross-reference)	
		Line Number	755
		Variable	756
		Subroutine	757
5.36	ATH3EDTP	EDIT THE CURRENT PAGE OF THE LEARNING MODULE	758
		107 To Initialize Pointers and Buffer	760
		108 To Input a Character and Verify for Validity	761
		109 To Insert Character into Buffer and Process Any Control Function	762
		110 To Transfer Character from Character Buffer to Storage Buffer; To Store Buffer on Disk When Full	763
		111 To Rubout Current Character	764
		112 To Underscore the Current Character	765
		113 To Allow the Storage Buffer to be Edited	766
		114 To Position Cursor Left or Right of Current Position	768
		115 To Position Cursor Up or Down of Current Position	769
		116 To Change Current Character Only	770
		117 To Delete a String of Characters	771
		118 To Insert Character from the Buffer	772
		119 To Position the Cursor at the Beginning or End of the Text Buffer	773
		120 To Compute the Cycle Number	774

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
		121 To Roll Screen When Position is Advanced Off of Screen	775
		123 To Set End of Data Flag and Save Buffer on Disk	776
		124 To Position Cursor and Buffer Pointer Such that the Next Character Entered Will Appear on Next Line	777
		MAPS (Cross-reference)	
		Line Number	778
		Variable	780
		Subroutine	781
6.0		FORMAT DATA SUBPROGRAMS	782
6.1	ATH1Ø1.Ø	FORMAT DEFINITION FOR THE INTRODUCTION PAGE	782
6.2	ATH1Ø2.0	FORMAT DEFINITION FOR THE IMPORTANCE OF THE PROGRAM TO YOU PAGE	784
6.3	ATH1Ø3.0	FORMAT DEFINITION FOR THE EXAMPLE OF HOW THE SYMBOLS ARE USED PAGE	786
6.4	ATH1Ø4.0	FORMAT DEFINITION FOR THE FOLLOWING PAGES: SYMBOL PRETEST (DIRECTIONS)	788
		Optional Criterion Test (Choose #1 or #2)	
		Optional Criterion Test (Directions)	
6.5	ATH1Ø4.1	FORMAT DEFINITION FOR THE FOLLOWING PAGES:	
		Optional Criterion Test (Symbols)	791
		Optional Criterion Test (Answers)	791
		Optional Criterion Test (End-of-Test)	791
6.6	ATH1Ø5.Ø	FORMAT DEFINITION FOR THE ORGANIZATION OF THE LEARNING PROGRAM PAGE	793
6.7	ATH1Ø6.Ø	FORMAT DEFINITION FOR THE SYMBOL LIST PAGE	795
6.8	ATH1Ø6.1	FORMAT DEFINITION FOR THE NOTE DIFFERENCES AMONG SYMBOLS (FLOWCHART) PAGE	797

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
6.9	ATH1Ø6.2	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Note Differences Among Symbols (Introduction). . . .	799
		Note Differences Among Symbols (Directions).	799
		Note Differences Among Symbols (Exercises)	799
		Note Differences Among Symbols (Answers)	799
6.10	ATH1Ø6.3	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Learn Symbol Definitions (Flowchart)	802
		Learn Symbol Definitions (Directions).	802
6.11	ATH1Ø6.4	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Learn Symbol Definitions (Definitions)	804
		Learn Symbol Definitions (Rest Directions)	804
6.12	ATH1Ø6.5	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Practice Defining the Symbols (Flowchart).	806
		Practice Defining the Symbols (Directions)	806
6.13	ATH1Ø6.6	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Practice Defining the Symbols (Exercises).	808
		Practice Defining the Symbols (Practice Directions).	808
		Practice Defining the Symbols (Rest Directions). . .	808
6.14	ATH1Ø6.7	FORMAT DEFINITION FOR THE TEST YOURSELF (FLOWCHART) PAGE	811
6.15	ATH1Ø6.8	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Test Yourself (Directions)	813
		Test Yourself (Memory Aids).	813
		Test Yourself (Answers).	813
6.16	ATH1Ø6.9	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Test Yourself (End of Test Directions)	815
		Test Yourself (Rest Directions).	815
6.17	ATH1Ø7.Ø	FORMAT DEFINITION FOR THE PRACTICE DEFINING ALL THE SYMBOLS (OVERVIEW AND DIRECTION) PAGE.	817

EXTENDED TABLE OF CONTENTS (continued)

VOLUME II

Section			Page
6.18	ATH1Ø7.1	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Practice Defining All the Symbols (Practice Exercises)	819
		Practice Defining All the Symbols (Answers)	819
6.19	ATH1Ø7.2	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Practice Defining All the Symbols (Ending Directions)	821
		Practice Defining All the Symbols (Practice Directions)	821
6.20	ATH1Ø8.Ø	FORMAT DEFINITION FOR THE CRITERION TEST (DIRECTIONS) PAGE	823
6.21	ATH1Ø8.1	FORMAT DEFINITIONS FOR THE FOLLOWING PAGES:	
		Criterion Test (Tests)	825
		Criterion Test (Answers)	825
		Criterion Test (End of Test Directions)	825
		Criterion Test (Congratulations)	825
6.22	ATH1Ø9.Ø	FORMAT DEFINITION FOR THE SUGGESTIONS FOR REFRESHER TRAINING PAGE	828
6.23	ATH11Ø.Ø	FORMAT DEFINITION FOR THE INDEX PAGE	830
7.Ø		FORMAT LISTINGS	832
8.Ø		COMPRESSING PROGRAM SOURCE CODE	878

LIST OF ILLUSTRATIONS

VOLUME II

Figure			Page
2.1	'AUTHOR'	AUTHOR SYSTEM INITIALIZING PROGRAM	41
3.1	'AUTHOR/1'	FORMAT DISPLAY PROGRAM	52
4.1	'AUTHOR/2'	DATA BASE MAINTENANCE PROGRAM.	119
4.2	'ATH2INIT'	INITIALIZE DATA BASE MAINTENANCE	120
4.3	'ATH2LOGC'	PROGRAM TO MAINTAIN DATA BASE ITEMS.	121
4.4	'ATH2LIST'	PROGRAM TO SORT SYMBOLS.	123
4.5	'ATH2LSTV'	PROGRAM TO DEFINE COMMON	124
4.6	'ATH2LSTP'	PROGRAM TO LIST DATA BASE ITEMS.	125
4.7	'AUTHOR/2'	ROUTINE 106--PROGRAM ROUTINE FOR ENTRY/EDIT OF TEXT.	127
4.8	'AUTHOR/2'	ROUTINE 109--PROGRAM ROUTINE FOR PROCESSING TEXT	128
5.1	'AUTHOR/3'	PROGRAM TO CREATE AND MAINTAIN LEARNING MODULE	337
5.2	'ATH3INIT'	PROGRAM TO INITIALIZE LEARNING MODULE MAINTENANCE.	338
5.3	'ATH3STAB'	PROGRAM TO LOAD SYMBOL TABLE INTO COMMON	339
5.4	'ATH3VTAB'	PROGRAM TO LOAD SYMBOL SORT PROGRAM.	339
5.5	'ATH3SRTS'	PROGRAM TO LOAD AND SORT SYMBOL TABLE.	340
5.6	'ATH3AUTO'	PROGRAM TO CONTROL THE CONSTRUCTION OF THE LEARNING MODULE.	341
5.7	'ATH3INDX'	PROGRAM TO INDEX THE LEARNING MODULE	344
5.8	'ATH3LIST'	PROGRAM TO INITIALIZE LIST PROCESSING BY LOADING NECESSARY COMMON VARIABLES	346

LIST OF ILLUSTRATIONS

VOLUME II

Figure			Page
5.9	'ATH3LSTV'	PROGRAM TO LOAD LISTING ROUTINES.	346
5.10	'ATH3LSTP'	PROGRAM FOR LISTING THE LEARNING MODULE	347
5.11	'ATH3LSTP'	ROUTINE 203--PROGRAM ROUTINE FOR THE RECALL AND EDIT OF LEARNING MODULE	348
8.1		COMBINATION OF MULTIPLE PROGRAM FILES	879

1.0 INTRODUCTION

Volume I of this report presented detailed operating instructions for the AUTHOR System.

This present volume (Volume II) contains the necessary information for a detailed examination of the design of the AUTHOR System programs and will be useful in modifying the programs to meet unique requirements of users. It includes the program listings and a summary of how the source programs are compressed to form programs that can be executed.

Additional assistance or access to a program in machine readable form will be provided by the Training Analysis and Evaluation Group (TAEG) to authorized agencies or individuals.

For information concerning the purpose of the AUTHOR System, see TAEG Report No. 58.[1]

1.1 ORGANIZATION OF THE REPORT

The remainder of this report is organized into seven sections. Section 2 provides a listing of the AUTHOR System Initializing Program. Section 3 presents the programs used to design the basic module formats. Section 4 presents the programs used to create and maintain the data base. Section 5 presents the programs used to create and maintain the output learning module. Sections 6 and 7 present the basic module formats, and section 8 provides a summary of how the source programs are compressed to form an executable set of programs.

The AUTHOR System programs make extensive use of program overlaying. This was necessary due to the hardware configuration. Therefore, the programs in this report are broken down by overlay and where necessary the overlay was broken down into parts. This added breakdown was necessary due to the large amount of program documentation.

Following each overlay and sub-overlay are two or three of the following reference maps:

- line number cross reference
- variable cross reference
- subroutine cross reference.

[1] Richard Braby, William F. Parrish, Charles R. Guitard, and James A. Aagard. Computer-Aided Authoring of Programmed Instruction for Teaching Symbol Recognition, TAEG Report No. 58. June 1978. Training Analysis and Evaluation Group, Orlando, FL.

2.0 AUTHOR SYSTEM INITIALIZING PROGRAM

The purpose of this 'AUTHOR' program is to initialize the disk device table and to allow the user to select the AUTHOR System program to be loaded and run. For operating instructions see section 4.0 in Volume I.

2.1 PROGRAM FLOWCHART

Figure 2.1 presents a program flowchart which shows the major steps of the initializing program.

When appropriate, the subroutine name or overlay name will be shown in the lower portion of the flowchart element. This indicates that the named subroutine or overlay performs the function described.

2.2 AUTHOR

The remainder of this section contains the program listing and reference maps.

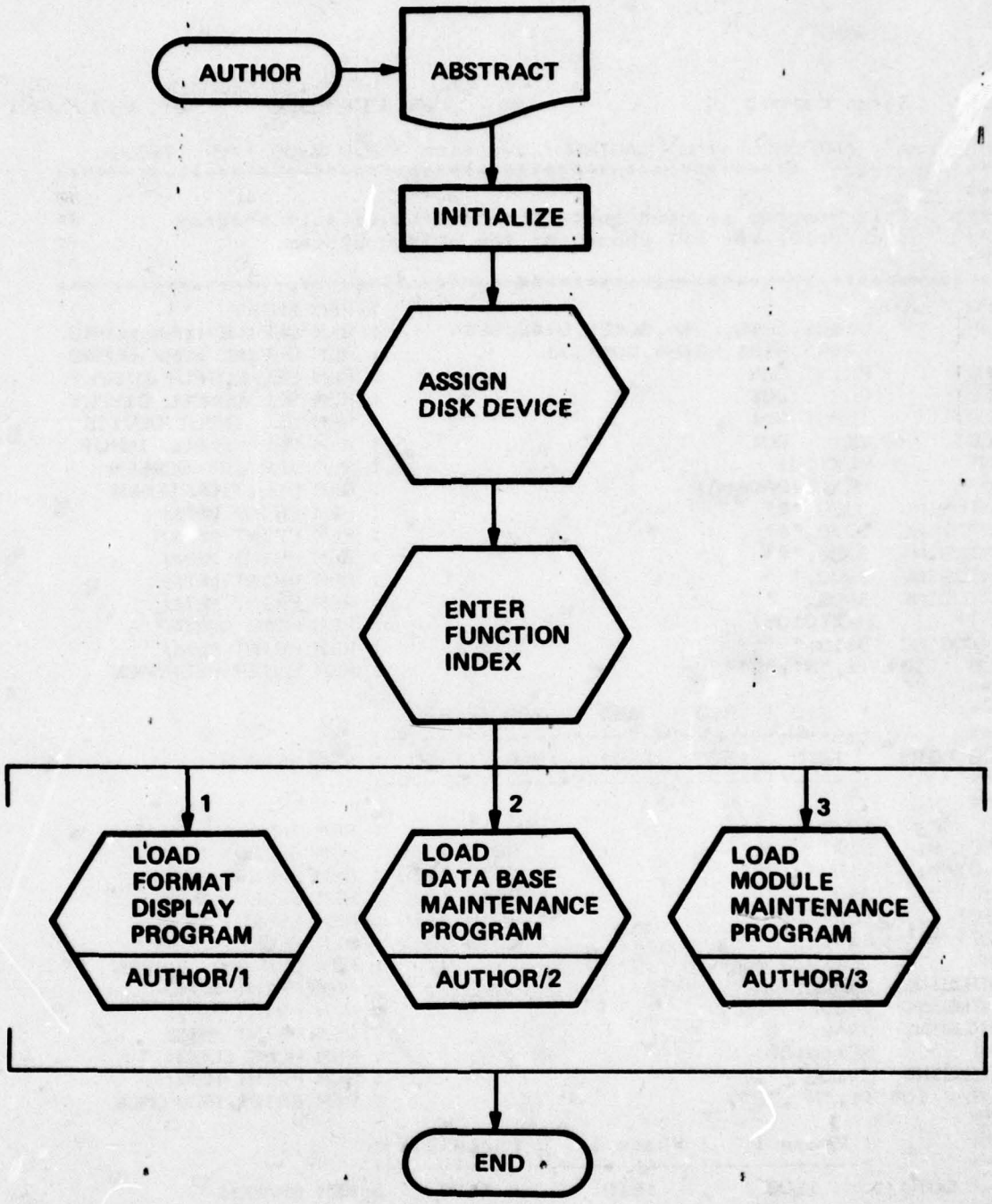


FIGURE 2.1. 'AUTHOR'--AUTHOR SYSTEM INITIALIZING PROGRAM

```
1000 %## Program > AUTHOR System  (AUTHOR)  Version > 2.0 2200 T/VP  780316
1010 REM **  Abstract  ************************************************************
1020 REM **                                                                       **
1030 REM **    This Program is used to initialize the default program             **
1040 REM **    disk drive for all phases of the AUTHOR System.                    **
1050 REM **                                                                       **
1060 REM ***************************************************************************
1070  DEFFN ' 000                                                : REM ENTRY
1080  DIM       L1$25,L2$8,L3$4,Q6$25,Q7$8,Q8$4                  : REM DEFINE WORK BUFRS
1090  DIM       R2$64,R1$1,R6$64,RO$(1)1                         : REM DEFINE WORK BUFRS
1100  SELECT    PRINT 005                                        : REM SEL OUTPUT DISPLY
1110  SELECT    CO    005                                        : REM SEL CONSOL DISPLY
1120  SELECT    INPUT 001                                        : REM SEL INPUT DEVICE
1130  SELECT    CI    001                                        : REM SEL CONSOL INPUT
1140  PRINT     HEX(03)                                          : REM CLR CRT SCREEN
1150  PRINT     HEX(010A0A0A)                                    : REM POSITION CUSOR
1160  PRINTUSING 3360,"#"                                        : REM PRINT MENU
1170  PRINTUSING 3370,"#"                                        : REM PRINT MENU
1180  PRINTUSING 3380,"#"                                        : REM PRINT MENU
1190  PRINTUSING 3390," "                                        : REM PRINT MENU
1200  PRINTUSING 3400," "                                        : REM PRINT MENU
1210  PRINT     HEX(0106)                                        : REM HOME CUSOR
1220  PRINTUSING 3410," ";                                       : REM PRINT MENU
1230  GOSUB ' 104 (1,"N","5")                                    : REM ENTER RESPONCE
1240 REM **
1250 REM **       !  310 !  B10 !  350 !  320 !  B20  !
1260 REM **       !------!------!------!------!-------!
1270  ON R2 GOTO    1310 , 1320 , 1330 , 1340 , 1350   : REM BRANCH
1280 REM **       !------!------!------!------!-------!
1290 REM **
1300  GOTO      1070                                             : REM INVALID VALUE
1310  SELECT #1 310                          : GOTO 1360 : REM SELECT DISK
1320  SELECT #1 B10                          : GOTO 1360 : REM SELECT DISK
1330  SELECT #1 350                          : GOTO 1360 : REM SELECT DISK
1340  SELECT #1 320                          : GOTO 1360 : REM SELECT DISK
1350  SELECT #1 B20                          : GOTO 1360 : REM SELECT DISK
1360  PRINT     HEX(030A0A0A)                                    : REM CLR CRT SCREEN
1370  PRINTUSING 3420," "                                        : REM PRINT MENU
1380  PRINTUSING 3430," "                                        : REM PRINT MENU
1390  PRINTUSING 3440," "                                        : REM PRINT MENU
1400  PRINT     HEX(0106)                                        : REM HOME CUSOR
1410  PRINTUSING 3450," ";                                       : REM PRINT MENU
1420  GOSUB ' 104 (1,"N","3")                                    : REM ENTER RESPONCE
1430 REM **
1440 REM **       ! Phase I   ! Phase II  ! Phase III !
1450 REM **       !-----------!-----------!-----------!
1460  ON R2 GOTO     1500    ,   1510    ,   1520    : REM BRANCH
1470 REM **       !-----------!-----------!-----------!
1480 REM **
1490  GOTO      1360                                             : REM INVALID BRANCH
1500  GOSUB ' 102 ("AUTHOR SYSTEM: Phase I","AUTHOR/1","0100")
1510  GOSUB ' 102 ("AUTHOR SYSTEM: Phase II","AUTHOR/2","0100")
1520  GOSUB ' 102 ("AUTHOR SYSTEM: Phase III","AUTHOR/3","0100")
1530  END                                                        : REM END OF PROGRAM
```

```
1540 REM ************************************************************************
1550 REM **                          LABELED  SUBROUTINES                      **
1560 REM ************************************************************************
1570 REM **
1580 REM **
1590 REM **   Subroutine > ********************************************* 1 0 1
1600 REM **   Function   >
1610 REM **     Cusor Address and Screen erase
1620 REM **   Varaibles  >
1630 REM **     Q1     > row position !  Q3     > lines to erase
1640 REM **     Q2     > col position !  Q4     > char(s) erase
1650 REM **     Q1$()  > cusor vert   !  Q2$()  > cusor horz
1660 REM **     Q3$()  > line ers     !  Q4$()  > micro cmnd bufr
1670 REM **     Q5     > line indx    !
1680 REM **
1690 REM ********************************************
1700 DEFFN ' 101   (Q1,Q2,Q3,Q4)                                  : REM ENTRY
1710 DIM           Q1$(24)1,Q2$(80)1,Q3$(80)1,Q4$(10)1            : REM DEFIN BUFRS
1720 INIT(0A)      Q1$()                                          : REM INIT ROW BUFR
1730 INIT(09)      Q2$()                                          : REM INIT COL BUFR
1740 INIT(20)      Q3$()                                          : REM INIT ERS BUFR
1750               Q1$(1) = HEX(01)                               : REM INIT TOP OF PAGE
1760               Q2$(1) = HEX(0D)                               : REM INIT BEG OF LINE
1770 SELECT        PRINT 405                                      : REM SEL OUTPUT DISPLY
1780 IF            Q3 = 0                        THEN 1890        : REM NO LINES TO ERASE
1790 IF            Q4 <> 0                       THEN 1810        : REM CHARS TO ERASE
1800               Q4 = 81 - Q2                                   : REM ERS REST OF LINE
1810 FOR           Q5 = 1 TO Q3                                   : REM ERASE Q3 LINES
1820 IF            Q5 <> 1                       THEN 1850        : REM NOT FIRST LINE
1830 $GIO ROW/005 (A000,Q4$())Q1$()<1,Q1>                         : REM POSITION CUSOR
1840 GOTO          1860                                           : REM BRANCH TO POS COL
1850 $GIO ROW/005 (A000,Q4$())Q1$()<2,1>                          : REM POSITION CUSOR LF
1860 $GIO COL/005 (A000,Q4$())Q2$()<1,Q2>                         : REM POSITION CUSOR
1870 $GIO ERS/005 (A000,Q4$())Q2$()<1,Q4>                         : REM ERASE TEXT
1880 NEXT          Q5                                             : REM CONT ERASING
1890 $GIO ROW/005 (A000,Q4$())Q1$()<1,Q1>                         : REM REPOSITION CUSOR
1900 $GIO COL/005 (A000,Q4$())Q2$()<1,Q2>                         : REM REPOSITION CUSOR
1910 SELECT        PRINT 005                                      : REM SEL OUTPUT DISPLY
1920 RETURN                                                       : REM EXIT
```

```
1930 REM **
1940 REM **
1950 REM **   Subroutine > ********************************************  1 0 2
1960 REM **   Function   >
1970 REM **     To load an over lay Routine.
1980 REM **   Variables  >
1990 REM **     Q6$() > description        !  Q7$() program name
2000 REM **     Q8$() > begin line number !
2010 REM **
2020 REM ***********************************************
2030 DEFFN ' 102  (Q6$,Q7$,Q8$)                          : REM ENTRY
2040 SELECT       PRINT 005(80)                          : REM SEL OUTPUT DISPLY
2050 PRINT        HEX(03)                                : REM CLEAR CRT
2060 PRINTUSING   3340," "                               : REM PRINT TITLE
2070 PRINT
2080 PRINTUSING   3350," "                               : REM PRINT SUBTITLE
2090 GOSUB ' 101  (08,06,0,0)                            : REM POSITION CUSUR
2100 PRINT   "System Now Loading > ";Q6$;" Routines"     : REM PRINT DESCRIPTION
2110 GOSUB ' 101  (10,06,0,0)                            : REM POSITION CUSUR
2120 PRINT   "Program being Loaded > ";Q7$               : REM PRINT PROG NAME
2130 IF           Q8$   <> "0100"            THEN 2150   : REM LOAD AT LINE 0100
2140 LOAD         DC T #1, Q7$   100
2150 IF           Q8$   <> "2000"            THEN 2170   : REM LOAD AT LINE 2000
2160 LOAD         DC T #1, Q7$   1970
2170 IF           Q8$   <> "4000"            THEN 2190   : REM LOAD AT LINE 4000
2180 LOAD         DC T #1, Q7$   4000
2190 IF           Q8$   <> "6000"            THEN 2210   : REM LOAD AT LINE 6000
2200 LOAD         DC T #1, Q7$   3110
2210 GOSUB ' 101  (20,1,0,0)                             : REM POSITION CUSUR
2220 PRINT        "INVALID PROGRAM SPECIFICATION"
2230 END                                                 : REM END PROG EXECUTION
```

```
2240  REM **
2250  REM **
2260  REM **  Subroutine  ********************************************** 1 0 4
2270  REM **  Abstract >
2280  REM **    Accept ALPHA and NUMERIC Input.
2290  REM **  Varaibles  >
2300  REM **    RO$() > entry buffer      ! R1  > max length
2310  REM **    R1$   > 'A' or 'N' or 'E'! R2  > return <num>
2320  REM **    R2$   > return <alph>     ! R3  > char pointer
2330  REM **    R6$   > initial value     ! R9  > scratch count
2340  REM **
2350  REM ***********************************************
2360  DEFFN ' 104   (R1,R1$,R6$)                                  : REM ENTRY
2370  DIM           R9$(1)64,R8$(9)24,R4$(1)10                    : REM DEFINE WORK BUFRS
2380                R4$(1)=HEX(010802A7030D03E62020)              : REM DEFIN CHAR TRANS
2390                R8$(9)=HEX(05163F20556677000000)              : REM DEFIN MICRO CMNDS
2400                R8$(1)=HEX(400642604008121142504008113614541608114G)
2410  INIT(3F)      R9$()                                         : REM INIT CHAR BUFR
2420  $GIO PRINT    /005 (A000,R8$(9))R9$()<1,R1>                 : REM PRNT CUSOR BKGRND
2430  INIT(08)      R9$()                                         : REM INIT CHAR BUFR
2440  $GIO PRINT    /005 (A000,R8$(9))R9$()<1,R1>                 : REM PRNT CUSOR BKGRND
2450  $GIO          /005 (1136,R8$(9))                            : REM INIT REGISTER
2460                R9  =  1                                      : REM PRNT POS INDX
2470                R3  =  1                                      : REM BUFR POS INDX
2480  INIT          (20) R2$                                      : REM CLR BUFR
2490  REM **
2500  KEYIN         RO$(1), 2500, 2500                            : REM CLR INPUT BUFR
2510                STR(R8$(9),5,1) = STR(R2$,R9,1)               : REM STOR OUT CHAR
2520  IF            R1$ <> "E"                    THEN 2540       : REM ECHO CHARACTR
2530                STR(R8$(9),5,1) = HEX(20)                     : REM SET NO ECHO
2540  KEYIN         RO$(1), 2640, 2580                            : REM INPUT CHAR
2550  $GIO          INPUT WAIT/005(R8$(1),R8$(9))                 : REM PRNT CUSOR
2560  GOTO          2540                                          : REM CONT INPT WAIT
2570  REM **
2580  PRINT         HEX(0808);                                    : REM FNT KEY ENTRY
2590  INPUT         R2$                                           : REM REENTER SPL KEY
2600                R3 = 5                                        : REM SET LENGTH
2610                STR(R2$,R3) = HEX(20)                         : REM CLR BUFR END
2620  GOTO          3190                                          : REM CONV IN DATA
2630  REM **
2640  IF            RO$(1) < HEX(08)              THEN 2690       : REM INVALID INPT
2650  $TRAN         (RO$()<1,1>, R4$()<1,10>) R                   : REM CONV CTL CHARS
2660  IF            RO$(1) > HEX(7F)              THEN 2690       : REM INVALID INPT
2670  IF            RO$(1) > HEX(1F)              THEN 2740       : REM VALID CHAR
2680  IF            RO$(1) < HEX(04)              THEN 2740       : REM VALID CTL CHAR
2690  $GIO          TONE/005(R8$(7),R8$(9))                       : REM SOUND BELL
2700  GOTO          2540                                          : REM RE-ENTER CHAR
2710  REM **
2720  REM **                  ! RUBOUT ! UNDERSCORE ! ESCAPE !
2730  REM **                  !--------!------------!--------!
2740  ON VAL(RO$(1)) GOTO       2920   ,    3030    ,  3090   : REM BRANCH
2750  REM **                  !--------!------------!--------!
```

```
2760  REM **
2770  REM ********************************************** Enter Character
2780  REM **
2790  IF            R3 = R1 + 1                  THEN 2690  : REM BUFR FULL
2800                STR(R2$,R3,1) = RO$(1)                  : REM MOVE TO BUFR
2810                R3 = R3 + 1                             : REM INCR BUFR INDX
2820  IF            R3 = 2                       THEN 2850  : REM NOT FRST CHAR
2830                R9 = R9 + 1                             : REM INCR DSPLY INDX
2840  PRINT         HEX(09);                                : REM POSITION CUSOR
2850  IF            R1$ <> "E"                   THEN 2870  : REM ECHO CHARACTR
2860                RO$(1) = HEX(20)                        : REM SET NO ECHO
2870  PRINT         RO$(1);HEX(08);                         : REM PRNT CHAR
2880  GOTO          2510                                    : REM FECTH NXT CHAR
2890  REM **
2900  REM ********************************************** Rubout Character
2910  REM **
2920  IF            R3 = 1                       THEN 2510  : REM BUFR EMPTY
2930  PRINT         HEX(3F08);                              : REM POSITION CUSOR
2940                R3 = R3 - 1                             : REM DCRMNT BUF INDX
2950                STR(R2$,R3,1) = HEX(20)                 : REM CLR FRM BUFR
2960  IF            R9 = 1                        THEN 2990 : REM CUSOR POSITIONED
2970                R9 = R9 - 1                             : REM DCRMNT DSP INDX
2980  PRINT         HEX(08);                                : REM POSITION CUSOR
2990  GOTO          2510                                    : REM FETCH NXT CHAR
3000  REM **
3010  REM ********************************************** Underscore char.
3020  REM **
3030  AND           (STR(R2$,R9,1),7F)                      : REM MASK CHAR
3040  ADD           (STR(R2$,R9,1),80)                      : REM UNDERSCORE
3050  GOTO          2510                                    : REM FETCH NXT CHAR
3060  REM **
3070  REM ********************************************** String Entered
3080 REM **
3090  IF            R3 <> 1                      THEN 3160 : REM BUFR NOT EMPTY
3100                R2$ = R6$                               : REM INSRT DFLT VAL
3110  IF            R1$ <> "E"                   THEN 3130 : REM ECHO DFLT VAL
3120                R6$ = " "                               : REM CLR DFLT VAL
3130  PRINT         HEX(08);                                : REM POSITION CUSOR
3140  GOTO          3180                                    : REM BRNCH TO PRNT
3150  REM **
3160                STR(R2$,R3) = HEX(20)                   : REM CLR REST OF BUF
3170                R6$ = R2$                               : REM MOVE TO PRNT BUF
3180  PRINT         HEX(09);STR(R6$,R3,ABS(R1-R3)+1);       : REM PRNT STRNG
3190  IF            R1$ <> "N"                   THEN 3280 : REM RET ALPHA VAL
3200  IF            NUM(R2$) = 64                THEN 3270 : REM VALID NUM FMT
3210                R2  =  R1                               : REM SET CUSOR INDX
3220  IF            R3 <> R1+1                   THEN 3240 : REM INDX IS OK
3230                R2 =  R1+2                              : REM RESET INDX
3240  INIT(08) R9$(): $GIO/005(A000,R8$(9))R9$()<1,R2>      : REM PRNT CUSOR BKGRND
3250  $GIO          TONE/005(R8$(7),R8$(9))                 : REM ERR TONE
3260  GOTO          2410                                    : REM REENTER STRNG
3270  CONVERT       R2$ TO R2                               : REM SET NUMERIC VAL
3280  RETURN                                                : REM EXIT
```

```
3290 REM **
3300 REM **
3310 REM ************************************************************************
3320 REM **                          OUTPUT IMAGE STATEMENTS                     **
3330 REM ************************************************************************
3340 %     A U T H O R    An  Automatic Authoring System #
3350 %           * *  Structure Loading  * *             #
3360 % DISK MENU                1 - Flopy Disk Drive  #1      <310>
3370 %                          2 - Flopy Disk Drive  #2      <B10>
3380 %                          3 - Flopy Disk Drive  #3      <350>
3390 %                          4 - Fixed Disk Plater       # <320>
3400 %                          5 - Removable Disk Plater   # <B20>
3410 % Enter Code of the Disk Containing AUTHOR Programs > #
3420 % LOADING MENU # 1 - Construct Basic Module Formats   (AUTHOR/1)
3430 %              # 2 - Create/Maintain Data Base        (AUTHOR/2)
3440 %              # 3 - Create/Maintain Learning Module (AUTHOR/3)
3450 % Enter Code of the Function You wish to Perform > #
```

AUTHOR System: AUTHOR - # MAP

```
(0010) - 0001
(0100) - 2140
 1070  - 1300
 1310  - 1270
 1320  - 1270
 1330  - 1270
 1340  - 1270
 1350  - 1270
 1360  - 1310 1320 1330 1340 1350 1490
 1500  - 1460
 1510  - 1460
 1520  - 1460
 1810  - 1790
 1850  - 1820
 1860  - 1840
 1890  - 1780
 1970  - 2160
 2150  - 2130
 2170  - 2150
 2190  - 2170
 2210  - 2190
 2410  - 3260
 2500  - 2500
 2510  - 2880 2920 2990 3050
 2540  - 2520 2500 2700
 2580  - 2540
 2640  - 2540
 2690  - 2640 2660 2790
 2740  - 2670 2680
 2850  - 2820
 2870  - 2850
 2920  - 2740
 2990  - 2960
 3030  - 2740
 3090  - 2740
 3110  - 2200
 3130  - 3110
 3160  - 3090
 3180  - 3140
 3190  - 2620
 3240  - 3220
 3270  - 3200
 3280  - 3190
 3340  - 2060
 3350  - 2080
 3360  - 1160
 3370  - 1170
 3380  - 1180
 3390  - 1190
 3400  - 1200
 3410  - 1220
 3420  - 1370
 3430  - 1380
 3440  - 1390
 3450  - 1410
(4000) - 2180
(9999) - 0001
```

```
L1$   - 1080
L2$   - 1080
L3$   - 1080
Q1    - 1700 1830 1890
Q1$(  - 1710 1720 1750 1830 1850 1890
Q2    - 1700 1800 1860 1900
Q2$(  - 1710 1730 1760 1860 1900
Q3    - 1700 1780 1810
Q3$(  - 1710 1740 1870
Q4    - 1700 1790 1800 1870
Q4$(  - 1710 1830 1850 1860 1870 1890 1900
Q5    - 1810 1820 1880
Q6$   - 1080 2030 2100
Q7$   - 1080 2030 2120 2140 2160 2180 2200
Q8$   - 1080 2030 2130 2150 2170 2190
R0$(  - 1090 2500 2540 2640 2650 2660 2670 2680 2740 2800 2860 2870
R1    - 2360 2420 2440 2790 3180 3210 3220 3230
R1$   - 1090 2360 2520 2850 3110 3190
R2    - 1270 1460 3210 3230 3240 3270
R2$   - 1090 2480 2510 2590 2610 2800 2950 3030 3040 3100 3160 3170 3200 3270
R3    - 2470 2600 2610 2790 2800 2810 2820 2920 2940 2950 3090 3160 3180 3220
R4$(  - 2370 2380 2650
R6$   - 1090 2360 3100 3120 3170 3180
R8$(  - 2370 2390 2400 2420 2440 2450 2510 2530 2550 2690 3240 3250
R9    - 2460 2510 2830 2960 2970 3030 3040
R9$(  - 2370 2410 2420 2430 2440 3240
```

```
1070 DEFFN' 0
1700 DEFFN' 101
       - 2090 2110 2210
2030 DEFFN' 102
       - 1500 1510 1520
2360 DEFFN' 104
       - 1230 1420
```

3.0 FORMAT DISPLAY PROGRAM

The purpose of this program is to aid the programmer who wishes to modify the formats of the AUTHOR System. This program will allow the programmer to display the format on one of three output devices. Then after modifying the data subprogram, the programmer may re-display the format. The name of this program is 'AUTHOR/1'. For operating instructions see section 6.0 in Volume I.

3.1 PROGRAM FLOWCHART

Figure 3.1 presents a program flowchart which shows the major steps of the display program. When appropriate the subroutine name or overlay name will be shown in the lower portion of the flowchart element. This indicates that the named routine will perform the function described.

3.2 AUTHOR/1

The 'AUTHOR/1' program is presented in five parts due to the amount of documentation contained within the program.

3.2.1 ATH1/I

This section of the 'AUTHOR/1' program contains the main control logic along with subroutines.

The remainder of this section contains the 'ATH1/I' program listing and reference maps.

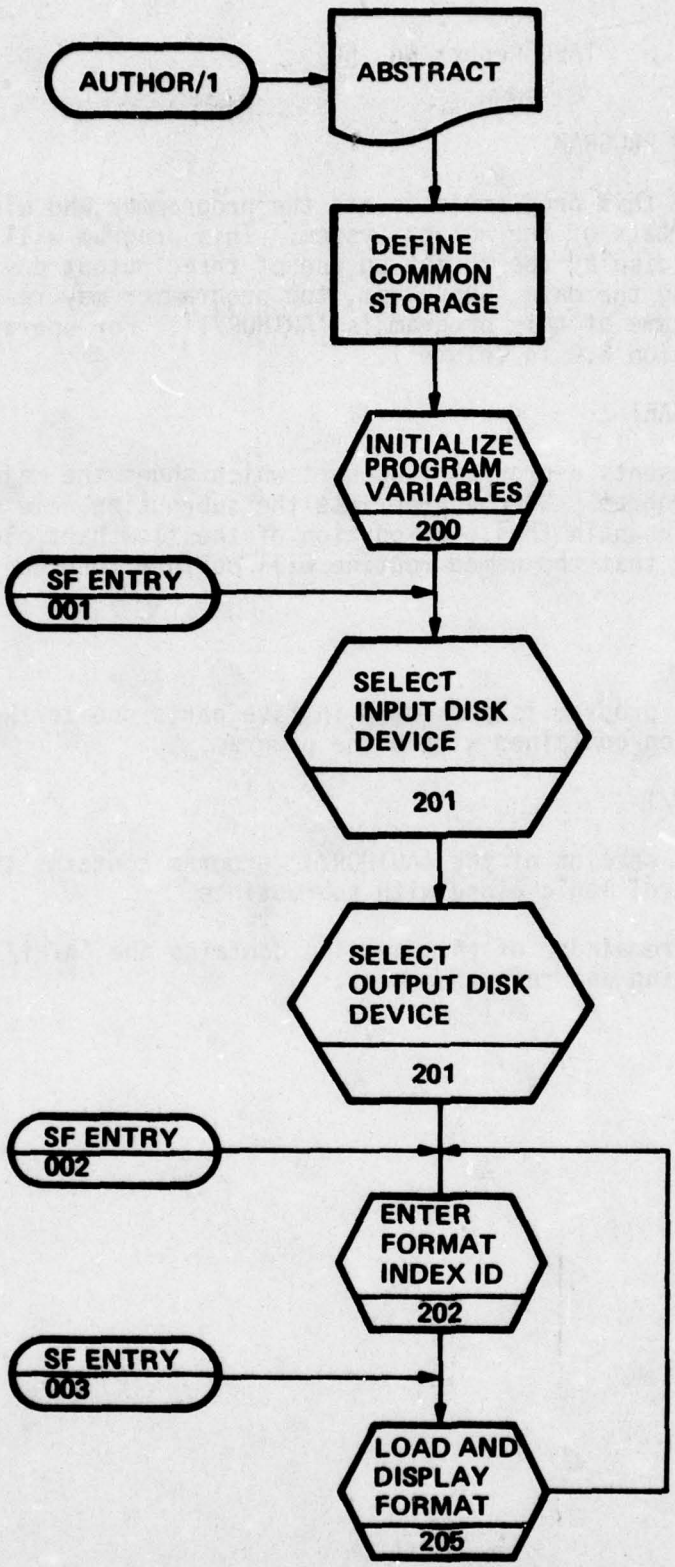


FIGURE 3.1. 'AUTHOR/1' FORMAT DISPLAY PROGRAM

```
AUTHOR System: Phase 1 - ATH1/I     - START

100  %## Program > AUTHOR/1   (ATH1/I  )   Version > 2.0 2200 T/VP   780316
104  REM **
108  REM **
112  REM **  Abstract  ************************************************
116  REM **                                                           **
120  REM **    Phase I of the AUTHOR programming system is used       **
124  REM **    create and maintain the basic module format Data Base  **
128  REM **                                                           **
132  REM ***************************************************************
136  REM **
140  REM **
144  COM          R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,01     : REM DEFINE VARIABLES
148  COM          R0$2,R1$2,R2$64,R3$8,R4$64,L1$4,L2$4
152  COM          R5$1,R6$64,R7$10,R8$2,R9$64
156  COM          R0$(1)1,R2$(2)2,R4$(20)16,R5$(10)1
160  COM          R6$(16)1,R7$(64)1,R8$(9)24,R9$(1)64
164  COM          Q1$(1)2,Q2$(16,64)1,S4$(1)64
168  COM          L1$25,L2$8,L3$4,K5$(1)64
172  COM          W2$9,W3$9,W4$13,W5$12,W6$16
176 REM **
180 REM **
184  SELECT       PRINT 405                                : REM SEL OUTPUT DEV
188  GOSUB ' 200                                           : REM INIT TABLES
192  DEFFN ' 001                                           : REM FUNCTION INTRUPT
196  GOSUB ' 201                                           : REM SEL I/O DEVICES
200  DEFFN ' 002                                           : REM FUNCTION INTRUPT
204  GOSUB ' 202                                           : REM SEL FORMAT FILE
208  DEFFN ' 003                                           : REM FUNCTION INTRUPT
212  GOSUB ' 205 (01)                                      : REM DISPLAY FORMAT
216  GOTO    204                                           : REM LOAD NEXT FILE
220 REM **
224 REM **
228  DEFFN ' 004  HEX(4C495354205320)                      : REM LIST S
232  DEFFN ' 005  HEX(534352415443485423312C22)            : REM SCRATCH T#1
236  DEFFN ' 006  HEX(5341564544435424282C22)              : REM SAVEDCT(
240  DEFFN ' 007  HEX(22292331 2C22)                       : REM )#1,
244  DEFFN ' 016  HEX(30312E300D)                          : REM 01.0
248  DEFFN ' 017  HEX(30322E300D)                          : REM 02.0
252  DEFFN ' 018  HEX(30332E300D)                          : REM 03.0
256  DEFFN ' 019  HEX(30342E300D)                          : REM 04.0
260  DEFFN ' 020  HEX(30352E300D)                          : REM 05.0
264  DEFFN ' 021  HEX(30362E300D)                          : REM 06.0
268  DEFFN ' 022  HEX(30372E300D)                          : REM 07.0
272  DEFFN ' 023  HEX(30382E300D)                          : REM 08.0
276  DEFFN ' 024  HEX(30392E300D)                          : REM 09.0
280  DEFFN ' 025  HEX(31302E300D)                          : REM 10.0
284  DEFFN ' 026  HEX(31312E300D)                          : REM 11.0
288  DEFFN ' 027  HEX(A5434F3231353AA0D2304329 AA5434F30303 50D): REM TOP PAG
292  DEFFN ' 028  HEX(41544831)                            : REM ATH1
296  DEFFN ' 029  HEX(415554484F522F31)                    : REM AUTHOR/1
300  DEFFN ' 030  HEX(31302C3930303000D)                   : REM 10,9000
304  DEFFN ' 031  HEX(39303030 2C3939393 90D)              : REM 9000,9999
308  DEFFN        R(R) = 256*R + VAL(STR(R2$(1),2,1))      : REM BIN TO DEC CONV
```

```
480  REM **
484  REM **
488  REM       Subroutine > ********************************************** 1 0 1
492  REM       Function   >
496  REM        Cusor Address and Screen erase
500  REM       Varaibles  >
504  REM        K1     > row position !  K3     > lines to erase
508  REM        K2     > col position !  K4     > char(s) erase
512  REM        R6$()  > cusor vert   !  R7$()  > cusor horz
516  REM        K5,K5$() > line ers !  R8$()  > micro cmnd bufr
520  REM
524  RLM ********************************************
528  DEFFN '  101   (K1,K2,K3,K4)                                  : REM ENTRY
532  IF             K3 = 0                          THEN 576      : REM ERASE NO LINE
536  IF             K4 <> 0                         THEN 544      : REM CHAR TO ERS SPCFD
540                 K4 = 65 - K2                                  : REM ERS REST OF LINE
544  FOR            K5 = 1 TO K3                                  : REM BEGIN LOOP K5
548  IF             K5 <> 1                         THEN 560      : REM NOT FIRST LINE
552  $GIO ROW       /005(A000  ,R8$(9))R6$()<1,K1>                : REM POSITION CUSOR
556                 GOTO 564                                      : REM BRANCH TO POS COL
560  $GIO ROW       /005(A000  ,R8$(9))R6$()<2,1>                 : REM POSITION CUSUR LF
564  $GIO COL       /005(A000  ,R8$(9))R7$()<1,K2>                : REM POSITION CUSUR
568  $GIO ERASE     /005(A000  ,R8$(9))K5$()<1,K4>                : REM ERASE TEXT
572  NEXT           K5                                            : REM CONTINUE LOOP K5
576  $GIO ROW       /005(A000  ,R8$(9))R6$() <1,K1>               : REM REPOSITION CUSOR
580  $GIO COL       /005(A000  ,R8$(9))R7$() <1,K2>               : REM REPOSITION CUSOR
584  RETURN                                                       : REM EXIT
```

```
588  REM **
592  REM **
596  REM      Subroutine > ********************************************** 1 0 2
600  REM      Function   >
604  REM        To load an over lay Routine.
608  REM      Variables  >
612  REM        L1$ > description       !  L3$ > begining line
616  REM        L2$ > overlay name      !
620  REM
624  REM ******************************************
628  DEFFN '  102  (L1$,L2$,L3$)                          : REM ENTRY
632  PRINT         HEX(03)                               : REM CLEAR CRT
636  PRINT         "A U T H O R   S Y S T E M"           : REM PRINT TITLE
640  PRINT         HEX(0D0A);"      P H A S E   III"     : REM PRINT SUBTITLE
644  GOSUB '  101  (5,10,0,0)                            : REM POSITION CUSOR
648  PRINT    "System Now Loading >>> ";L1$;" Routine":  REM PRINT DESCRIPTION
652  GOSUB '  101  (7,10,0,0)                            : REM POSITION CUSOR
656  PRINT         "Program being Loaded >>> ";L2$       : REM PRINT PROG NAME
660  IF            L3$ <> "0100"                THEN 668 : REM LOAD AT LINE 0100
664  LOAD          DC T #1, L2$ 100
668  IF            L3$ <> "1500"                THEN 676 : REM LOAD AT LINE 1500
672  LOAD          DC T #1, L2$ 1552
676  IF            L3$ <> "4000"                THEN 684 : REM LOAD AT LINE 4000
680  LOAD          DC T #1, L2$ 4000
684  IF            L3$ <> "6000"                THEN 692 : REM LOAD AT LINE 6000
688  LOAD          DC T #1, L2$ 6000
692  IF            L3$ <> "7000"                THEN 700 : REM LOAD AT LINE 7000
696  LOAD          DC T #1, L2$ 7000
700  IF            L3$ <> "9000"                THEN 708 : REM LOAD AT LINE 9000
704  LOAD          DC T #1, L2$ 9000
708  GOSUB ' 000                                         : REM ERR PROCESSING
712  RETURN                                              : REM EXIT
```

```
716  REM **
720  REM **
724  REM **  Subroutine  ********************************************** 1 0 4
728  REM **  Abstract >
732  REM **    Accept ALPHA and NUMERIC Input.
736  REM **  Varaibles  >
740  REM **    R0$() > entry buffer      ! R1  > max length
744  REM **    R1$   > 'A' or 'N' or 'E' ! R2  > return <num>
748  REM **    R2$   > return <alph>     ! R3  > char pointer
752  REM **    R6$   > initial value     ! R9  > scratch count
756  REM **
760  REM **********************************************
764  DEFFN ' 104  (R1,R1$,R6$)                              : REM ENTRY
768  INIT(3F)     R9$()                                     : REM INIT CHAR BUFR
772  $GIO PRINT   /005 (A000,R8$(9))R9$()<1,R1>             : REM PRNT CUSOR BKGRND
776  INIT(08)     R9$()                                     : REM INIT CHAR BUFR
780  $GIO PRINT   /005 (A000,R8$(9))R9$()<1,R1>             : REM PRNT CUSOR BKGRND
784  $GIO         /005 (1136,R8$(9))                        : REM INIT REGISTER
788               R9  =  1                                  : REM PRNT POS INDX
792               R3  =  1                                  : REM BUFR POS INDX
796  INIT         (20) R2$                                  : REM CLR BUFR
800  REM **
804  KEYIN        R0$(1), 804, 804                          : REM CLR INPUT BUFR
808               STR(R8$(9),5,1) = STR(R2$,R9,1)           : REM STOR OUT CHAR
812  IF           R1$ <> "E"                 THEN 820       : REM ECHO CHARACTR
816               STR(R8$(9),5,1) = HEX(20)                 : REM SET NO ECHO
820  KEYIN        R0$(1), 860, 836                          : REM INPUT CHAR
824  $GIO         INPUT WAIT/005(R8$(1),R8$(9))             : REM PRNT CUSOR
828  GOTO         820                                       : REM CONT INPT WAIT
832  REM **
836  PRINT        HEX(0808);                                : REM FNT KEY ENTRY
840  INPUT        R2$                                       : REM REENTER SPL KEY
844               R3 = 5                                    : REM SET LENGTH
848               STR(R2$,R3) = HEX(20)                     : REM CLR BUFR END
852  GOTO         1080                                      : REM CONV IN DATA
856  REM **
860  IF           R0$(1) < HEX(08)           THEN 880       : REM INVALID INPT
864  $TRAN       (R0$()<1,1>, R4$()<161,10>) R              : REM CONV CTL CHARS
868  IF          R0$(1) > HEX(7F)            THEN 880       : REM INVALID INPT
872  IF          R0$(1) > HEX(1F)            THEN 900       : REM VALID CHAR
876  IF          R0$(1) < HEX(04)            THEN 900       : REM VALID CTL CHAR
880  $GIO        TONE/005(R8$(7),R8$(9))                    : REM SOUND BELL
884  GOTO        820                                        : REM RE-ENTER CHAR
888  REM **
892  REM **                  ! RUBOUT ! UNDERSCORE ! ESCAPE !
896  REM **                  !--------!------------!--------!
900  ON VAL(R0$(1)) GOTO       972    ,    1016     ,  1040  : REM BRANCH
904  REM **                  !--------!------------!--------!
```

```
920  IF          R3 = R1 + 1                    THEN 880  : REM BUFR FULL
924              STR(R2$,R3,1) = RO$(1)                   : REM MOVE TO BUFR
928              R3 = R3 + 1                              : REM INCR BUFR INDX
932  IF          R3 = 2                         THEN 944  : REM NOT FRST CHAR
936              R9 = R9 + 1                              : REM INCR DSPLY INDX
940  PRINT       HEX(09);                                 : REM POSITION CUSOR
944  IF          R1$ <> "E"                     THEN 952  : REM ECHO CHARACTR
948              RO$(1) = HEX(20)                         : REM SET NO ECHO
952  PRINT       RO$(1);HEX(08);                          : REM PRNT CHAR
956  GOTO        808                                      : REM FECTH NXT CHAR
960  REM **
964  REM ********************************************** Rubout Character
968  REM **
972  IF          R3 = 1                         THEN 808  : REM BUFR EMPTY
976  PRINT       HEX(3F08);                               : REM POSITION CUSOR
980              R3 = R3 - 1                              : REM DCRMNT BUF INDX
984              STR(R2$,R3,1) = HEX(20)                  : REM CLR FRM BUFR
988  IF          R9 = 1                         THEN 1000 : REM CUSOR POSITIONED
992              R9 = R9 - 1                              : REM DCRMNT DSP INDX
996  PRINT       HEX(08);                                 : REM POSITION CUSOR
1000  GOTO        808                                     : REM FETCH NXT CHAR
1004  REM **
1008  REM ********************************************** Underscore char.
1012  REM **
1016  AND         (STR(R2$,R9,1),7F)                       : REM MASK CHAR
1020  ADD         (STR(R2$,R9,1),80)                       : REM UNDERSCORE
1024  GOTO        808                                      : REM FETCH NXT CHAR
1028  REM **
1032  REM ********************************************** String Entered
1036 REM **
1040  IF          R3 <> 1                        THEN 1068 : REM BUFR NOT EMPTY
1044              R2$ = R6$                                : REM INSRT DFLT VAL
1048  IF          R1$ <> "E"                     THEN 1056 : REM ECHO DFLT VAL
1052              R6$ = " "                                : REM CLR DFLT VAL
1056  PRINT       HEX(08);                                 : REM POSITION CUSOR
1060  GOTO        1076                                     : REM BRNCH TO PRNT
1064  REM **
1068              STR(R2$,R3) = HEX(20)                    : REM CLR REST OF BUF
1072              R6$ = R2$                                : REM MOVE TO PRNT BUF
1076  PRINT       HEX(09);STR(R6$,R3,ABS(R1-R3)+1);        : REM PRNT STRNG
1080  IF          R1$ <> "N"                     THEN 1116 : REM RET ALPHA VAL
1084  IF          NUM(R2$) = 64                  THEN 1112 : REM VALID NUM FMT
1088              R2  =  R1                                : REM SET CUSOR INDX
1092  IF          R3 <> R1+1                     THEN 1100 : REM INDX IS OK
1096              R2 =  R1+2                               : REM RESET INDX
1100  INIT(08) R9$(): $GIO/005(A000,R8$(9))R9$()<1,R2>    : REM PRNT CUSOR BKGRND
1104  $GIO        TONE/005(R8$(7),R8$(9))                  : REM ERR TONE
1108  GOTO        768                                      : REM REENTER STRNG
1112  CONVERT     R2$ TO R2                                : REM SET NUMERIC VAL
1116  RETURN                                               : REM EXIT
```

```
1120 REM **
1124 REM **
1128 REM **  Subroutine  ********************************************** 1 0 8
1132 REM **    Function   >
1136 REM **      To Output page buffer onto the selected
1140 REM **      output device.
1144 REM **      Output index code      1 - CRT
1148 REM **                             2 - TTY
1152 REM **                             3 - LPT
1156 REM **                             4 - END OF LPT FILE
1160 REM **    Variables  >
1164 REM **      W6 > output device index     ! RO$() > input register
1168 REM **      W7 > undrscr lokup indx      ! W2$   > return scan flag
1172 REM **      W8 > bufer out lin indx      ! W3$   > return scan flag
1176 REM **                                   ! W4$   > return scan flag
1180 REM **
1184 REM ********************************************
1188  DEFFN ' 108 (W6)                                    : REM ENTRY
1192  GOSUB ' 123                                         : REM ADJUST TEXT BUFR
1196  GOSUB ' 124                                         : REM SEL CHAR SCAN
1200 REM **
1204 REM **       !   CRT   !   TTY   !   LPT   !   END   !
1208 REM **       !---------!---------!---------!---------!
1212  ON W6 GOTO     1232  ,  1276  ,  1352  ,  1392      : REM BRANCH
1216 REM **       !---------!---------!---------!---------!
1220 REM **
1224 REM **                                               OUTPUT TO CRT DEVICE
1228 REM **
1232  SELECT #5   005                                     : REM SEL OUTPUT DEV
1236  GOSUB ' 109                                         : REM CREATE CRT OVRLAY
1240  GOSUB ' 110                                         : REM SEL T OR VP DISPLY
1244  IF          W2$ = "VP SYSTEM"          THEN 1252    : REM VP SELECTED
1248  GOSUB ' 111                                         : REM DISPLAY OVRLAY
1252  GOSUB ' 112                                         : REM DISPLAY BUFR
1256  KEYIN       RO$(1), 1260, 1260       : GOTO 1244    : REM INPUT WAIT
1260  GOTO        1408                                    : REM BRANCH TO EXIT
1264 REM **
1268 REM **                                               OUTPUT TO TTY DEVICE
1272 REM **
1276  SELECT #5   011                                     : REM SEL OUTPUT DEV
1280  GOSUB ' 109                                         : REM CREATE TTY OVRLAY
1284  GOSUB ' 113                                         : REM DISPLY INSTRUCTNS
1288  GOSUB ' 114 (1)                                     : REM PRINT UPR HASH
1292              W7 = 1                                  : REM INIT TABLE INDX
1296              X3 = 1                                  : REM INIT TABLE INDX
1300              Y3 = 1                                  : REM INIT TABLE INDX
1304  FOR         W8 = 1 TO 32                            : REM BEGIN BUFR LOOP
1308  GOSUB ' 115                                         : REM SCAN UNDRSCR TBL
1312  IF          W3$ = "NULL SCAN"          THEN 1324    : REM NO UNDERSCORING
1316  GOSUB ' 116                                         : REM PRINT UNDERSCORING
1320  GOTO        1328                                    : REM BRNCH TO LIN INCR
1324  GOSUB ' 117                                         : REM PRINT TEXT LINE
1328  NEXT        W8                                      : REM INCR LINE INDX
1332  GOSUB ' 114 (2)                                     : REM PRINT LWR HASH
1336  GOTO        1408                                    : REM BRANCH TO EXIT
```

```
1340 REM **                                                    OUTPUT TO LPT DEVICE
1344 REM **
1348 REM **
1352  SELECT #5   015                                        : REM SEL OUTPUT DEV
1356  SELECT      TAPE 07B                                   : REM SEL TAPE DEV
1360  GOSUB ' 118                                            : REM SEL UNDRSCR MODE
1364  GOSUB ' 119 (1)                                        : REM PRINT UPR HASH
1368  FOR         W8 = 1 TO 32                               : REM BEGIN BUFR LOOP
1372  GOSUB ' 117                                            : REM PRINT TEXT LINE
1376  NEXT        W8                                         : REM INCR LINE INDX
1380  GOSUB ' 119 (2)                                        : REM PRINT LOWR HASH
1384  IF          W4$ = "NO UNDERSCORE"      THEN 1396       : REM NO UNDERSCORING
1388  IF          W4$ = "OVERLAY"            THEN 1396       : REM IF OVRLAY OUTPUT
1392  GOSUB ' 120                                            : REM PRINT OVERLAY
1396  GOTO        1408                                       : REM BRANCH TO EXIT
1400 REM **
1404 REM **
1408  RETURN                                                 : REM EXIT
```

```
1412 REM **
1416 REM **
1420 REM **  Subroutine  ********************************************** 1 0 9
1424 REM **  Abstract >
1428 REM **    To scan text buffer and create lookup table and
1432 REM **    overlay buffer for selected output devices.
1436 REM **  Variables >
1440 REM **    W2$() > output text buffer     ! Q2$() > crt overlay buffer
1444 REM **    W3$() > underscore lookup tbl  ! W4$() > conversion buffer
1448 REM **    X4 > end of loop indx (row)    ! X5 > end of loop indx (col)
1452 REM **    R4 > lookup tbl indx (row)     ! R5 > lookup tbl indx (col)
1456 REM **    R6 > linear buff position      ! R5$() > single char storage
1460 REM **    W6 > output device indx        ! W9 > size of text buffer
1464 REM **
1468 REM ***********************************************
1472  DEFFN ' 109                                          : REM ENTRY
1476  DIM          W2$(32,64)1,W3$(1,200)2,W9$(5)8         : REM DEFINE WORK BUFRS
1480  DIM          X3$(1,075)2,Y3$(1,075)2                 : REM DEFINE WORK BUFRS
1484               W9$(1) = HEX(5FA0)                      : REM INIT CONV VALUES
1488               W9$(2) = HEX(20800900)                  : REM INIT CONV VALUES
1492               W9$(3) = HEX(20802D812F825F83)          : REM INIT CONV VALUES
1496               W9$(4) = HEX(2000200120022003)          : REM INIT CONV VALUES
1500               W9$(5) = HEX(2020)                      : REM INIT CONV VALUES
1504               W9 = 2048                               : REM SET SIZE TO MAX
1508               R2$(1) = "Z"                            : REM DEFINE SPL CHAR 1
1512               R2$(2) = "O"                            : REM DEFINE SPL CHAR 2
1516  $TRAN        (W2$()<1,W9>, W9$()<1,8>) R             : REM MINIZE UNDRSCRS
1520  IF           W6 <> 1                    THEN 1532    : REM NOT CRT OUTPUT
1524  MAT COPY     W2$() <1,1024> TO Q2$()<1,1024>         : REM SAVE UPPR BUFR
1528               W9 = 1024                               : REM SET BUFR SIZE
1532  MAT SEARCH   W2$()<1,W9>, > R5$(3) TO W3$()          : REM CREAT LOKUP TBL 1
1536  IF           W6 = 1                     THEN 1544    : REM NOT TTY OUTPUT
1540  $TRAN        (W2$()<1,W9>, W9$()<1,8>) 7F R          : REM CLR HIGH BIT
1544  MAT SEARCH   W2$()<1,W9>, = R2$(1) TO X3$()          : REM CREAT LOKUP TBL 2
1548  MAT SEARCH   W2$()<1,W9>, = R2$(2) TO Y3$()          : REM CREAT LOKUP TBL 3
1552               W3$(1,200)   = HEX(0000)                : REM SET EOI BYTE
1556               X3$(1,075)   = HEX(0000)                : REM SET EOI BYTE
1560               Y3$(1,075)   = HEX(0000)                : REM SET EOI BYTE
1564  IF           W6$ <> "NULL"              THEN 1568    : REM NULL CHAR SCAN
1565               X3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
1566               Y3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
1567  GOTO         1644                                    : REM BRNCH TO BLD TBL
1568  IF           W6$ = "BOTH"               THEN 1580    : REM BEG BLDIN TBLS
1569  IF           W6$ = "ZEE"                THEN 1572    : REM BLD ONLY Z TBL
1570               X3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
1571  GOTO         1612                                    : REM BRNCH TO BLD TBL O
1572               Y3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
```

```
1576  REM **
1580  FOR          R4 = 1 TO 1                                  : REM BEGIN TBL SCAN ROW
1584  FOR          R5 = 1 TO 075                                : REM BEGIN TBL SCAN COL
1588  GOSUB ' 125 (R4,R5,1,075,X3$(R4,R5),HEX(81))              : REM SCAN LOKUP TABLE 1
1592  NEXT         R5                                           : REM CONT  TBL SCAN COL
1596  NEXT         R4                                           : REM CONT  TBL SCAN ROW
1600               X3$(X6,X7) = HEX(FFFF)                       : REM SET EOI BYTE
1604  IF           W6$ = "BOTH"                THEN 1616        : REM EXECUTE SCAN
1608  REM **
1612  IF           W6$ <> "ZERO"               THEN 1644        : REM NULL 'O' SCAN
1616  FOR          R4 = 1 TO 1                                  : REM BEGIN TBL SCAN ROW
1620  FOR          R5 = 1 TO 075                                : REM BEGIN TBL SCAN COL
1624  GOSUB ' 125 (R4,R5,1,075,Y3$(R4,R5),HEX(82))              : REM SCAN LOKUP TABLE 2
1628  NEXT         R5                                           : REM CONT  TBL SCAN COL
1632  NEXT         R4                                           : REM CONT  TBL SCAN ROW
1636               Y3$(X6,X7) = HEX(FFFF)                       : REM SET EOI BYTE
1640  REM **
1644  FOR          R4 = 1 TO 1                                  : REM BEGIN TBL SCAN ROW
1648  FOR          R5 = 1 TO 200                                : REM BEGIN TBL SCAN COL
1652  GOSUB ' 125 (R4,R5,1,200,W3$(R4,R5),HEX(83))              : REM SCAN LOKUP TABLE 0
1656  NEXT         R5                                           : REM CONT  TBL SCAN COL
1660  NEXT         R4                                           : REM CONT  TBL SCAN ROW
1664               W3$(X6,X7) = HEX(FFFF)                       : REM SET EOI BYTE
1668  REM **
1672  REM **       !   CRT  !   TTY  !   LPT  !
1676  REM **       !--------!--------!--------!
1680  ON W6 GOTO     1692  ,  1716  ,  1728                     : REM BRANCH
1684  REM **       !--------!--------!--------!
1688  REM **
1692  MAT COPY     Q2$() TO W2$()<1025,1024>                    : REM MOVE SAVED BUFR
1696  $TRAN        (W2$()<1,W9>, W9$()<9,8>) 80 R               : REM CONV OVRLAY BUFR
1700  MAT COPY     W2$()<1,1024> TO Q2$()                       : REM STOR OVRLAY BUFR
1704  MAT COPY     W2$()<1025,1024> TO W2$()<1,1024>            : REM UPDATE TEXT BUFR
1708  GOTO         1740                                         : REM BRANCH TO EXIT
1712  REM **
1716  $TRAN        (W2$()<1,W9>, W9$()<1,8>) 7F R               : REM CLR UNDRSCR FLGS
1720  GOTO         1740                                         : REM BRANCH TO EXIT
1724  REM **
1728  $TRAN        (W2$()<1,W9>, W9$()<17,24>) 83 R             : REM CONV OVRLAY BUFR
1732  GOTO         1740                                         : REM BRANCH TO EXIT
1736  REM **
1740  RETURN                                                    : REM EXIT
```

```
1744 REM **
1748 REM **
1752 REM **  Subroutine  ********************************************** 1 1 0
1756 REM **  Abstract >
1760 REM **    To select type of CRT display to be used.
1764 REM **  Variables >
1768 REM **    W2$ > return type flag       ! RO$() > input register
1772 REM **
1776 REM ************************************************
1780  DEFFN ' 110                                          : REM ENTRY
1784  IF          W2$ = "VP SYSTEM"          THEN 1840 : REM SKIP ROUTINE
1788  IF          W2$ = "T  SYSTEM"          THEN 1840 : REM SKIP ROUTINE
1792              W2$ = "T  SYSTEM"                    : REM SET FLAG
1796  PRINT       HEX(03)                              : REM CLR CRT SCREEN
1800  GOSUB ' 101 (5,1,0,0)                            : REM POSITION CUSOR
1804  SELECT       PRINT 005                           : REM SEL OUTPUT MODE
1808  PRINTUSING   1856, " "                           : REM PRINT MENU
1812  PRINTUSING   1860, " "                           : REM PRINT MENU
1816  SELECT       PRINT 405                           : REM SEL OUTPUT MODE
1820  GOSUB ' 101 (2,1,0,0)                            : REM POSITION CUSOR
1824  PRINTUSING  1864, " ";                           : REM PRINT PROMPT
1828  KEYIN       RO$(1), 1832, 1832       : GOTO 1828 : REM RESPONCE WAIT
1832  IF          RO$(1) = "2"               THEN 1840 : REM SKP TO EXIT
1836              W2$ = "VP SYSTEM"                    : REM SET RTRUN FLAG
1840  PRINT       HEX(03)                              : REM CLR CRT SCREEN
1844 REM **
1848 REM **                                              OUTPUT IMAGE FORMATS
1852 REM **
1856  % CRT DISPLAY MENU      1 -  WANG 2200 VP Display   #
1860  %                       2 -  WANG 2200 T  Display   #
1864  %  Enter Type of CRT Display to be Used > #
1868 REM **
1872 REM **
1876  RETURN                                             : REM EXIT
```

```
1880 REM **
1884 REM **
1888 REM **  Subroutine  ********************************************** 1 1 1
1892 REM **  Abstract >
1896 REM **    To Display CRT overlay buffer onto selected CRT
1900 REM **    display.
1904 REM **  Variables >
1908 REM **    Q2$() > overlay buffer       ! W9 > buffer out line indx
1912 REM **    R8$ > field format definition!
1916 REM **
1920 REM ************************************************
1924  DEFFN ' 111                                          : REM ENTRY
1928                R8$ = HEX(5102)                       : REM DEFINE PACK FMT
1932  $GIO HOME     #5 (4001          ,R8$(9))            : REM HOME CURSOR
1936  FOR           W9 = 1 TO 16                          : REM BEG BUFFR OUT
1940  $PACK         (F=R8$) STR(R8$(9),7,2) FROM W9       : REM STOR LIN COUNT
1944  $GIO PRINT    #5 (R8$(6),R8$(9))Q2$()<(W9-1)*64+1,64> : REM PRINT LINE
1948  NEXT          W9                                    : REM CONT BUFR OUT
1952  RETURN                                              : REM EXIT
```

```
1956 REM **
1960 REM **
1964 REM **  Subroutine  ********************************************** 1 1 2
1968 REM **  Abstract >
1972 REM **    To Display CRT text buffer on to selected CRT
1976 REM **    display.
1980 REM **  Variables >
1984 REM **    W2$() > overlay buffer        ! W9 > buffer out line indx
1988 REM **    R8$ > field format definition! R9$() > scratch buffer
1992 REM **
1996 REM ************************************************
2000  DEFFN ' 112                                          : REM ENTRY
2004              R8$ = HEX(5102)                         : REM DEFINE PACK FMT
2008              R9$(1) = "Touch RETURN"                 : REM INIT MESSAGE
2012  $GIO HOME   #5 (4001          ,R8$(9))              : REM HOME CUSUR
2016  FOR         W9 = 1 TO 16                            : REM BEG BUFFR OUT
2020  $PACK       (F=R8$) STR(R8$(9),7,2) FROM W9         : REM STOR LIN COUNT
2024  $GIO PRINT  #5 (R8$(6),R8$(9))W2$()<(W9-1)*64+1,64> : REM PRINT LINE
2028  NEXT        W9                                      : REM CONT BUFR OUT
2032  $GIO MESG   #5 (400D400EA000,R8$(9))R9$()<1,13>     : REM PRINT MESSAGE
2036  RETURN                                              : REM EXIT
```

```
2040 REM **
2044 REM **
2048 REM **  Subroutine  ********************************************** 1 1 3
2052 REM **  Abstract >
2056 REM **    To Display Instructions to set up TTY device.
2060 REM **  Variables >
2064 REM **    W5$ > return scan flag      ! W4$() > i/o buffer #1
2068 REM **    W9  > scratch loop indx     ! W5$() > i/o buffer #2
2072 REM **    R0$() > input register      !
2076 REM **
2080 REM ***********************************************
2084  DEFFN ' 113                                          : REM ENTRY
2088  DIM          X9$(10)45                               : REM DEFINE WORK BUFR
2092  IF           W5$ = "SKIP DISPLAY"      THEN 2236     : REM SKIP DISPLAY
2096               W5$ = "SKIP DISPLAY"                    : REM SET RETURN FLAG
2100  PRINT        HEX(03)                                 : REM CLR CRT SCREEN
2104  GOSUB ' 101  (5,5,0,0)                               : REM POSITION CUSOR
2108  PRINTUSING   2252, " "                               : REM PRINT INSTRUTN 1
2112  GOSUB ' 101  (7,5,0,0)                               : REM POSITION CUSOR
2116  PRINTUSING   2256, " "                               : REM PRINT INSTRUTN 2
2120  GOSUB ' 101  (9,5,0,0)                               : REM POSITION CUSOR
2124  PRINTUSING   2260, " "                               : REM PRINT INSTRUTN 3
2128  GOSUB ' 101  (16,1,0,0)                              : REM POSITION CUSOR
2132  PRINTUSING   2272, " ";                              : REM PRINT CONT PROMPT
2136  KEYIN        R0$(1), 2144, 2144      : GOTO 2136     : REM RESPONCE WAIT
2140 REM **
2144  $GIO READY   #5 (4500         ,R8$(9))               : REM SEND CTL CHAR
2148  $IF  ON      #5, 2172                                : REM DEVICE READY
2152  GOSUB ' 101  (11,10,0,0)                             : REM POSITION CUSOR
2156  PRINTUSING   2264, " "                               : REM PRINT ERR MSG
2160  $GIO PAUSE   /005(R8$(4),R8$(9))                     : REM ERROR TONE
2164  GOSUB ' 101  (11,10,1,0)                             : REM ERASE ERR MSG
2168  GOTO         2144                                    : REM CONT READY TEST
2172  PRINT        HEX(030A0A0A0A0A)                       : REM CLR CONSOLE DEV
2176  PRINTUSING   2268, " "                               : REM PRINT INSTRUCTN
2180 REM **
2184  GOSUB ' 121                                          : REM CLR/SET TAB STOPS
2188  SELECT       PRINT 211(100)                          : REM SEL CONSOLE DEVICE
2192  PRINTUSING   2276, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2196  PRINTUSING   2280, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2200  PRINTUSING   2284, " "          : PRINT HEX(0A0A)    : REM PRINT INSTRUCTNS
2204  PRINTUSING   2288, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2208  PRINTUSING   2292, " "          : PRINT HEX(0A0A)    : REM PRINT INSTRUCTNS
2212  PRINTUSING   2296, " ";                              : REM PRINT INSTRUCTNS
2216  PRINTUSING   2300, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2220  INIT(0A)     X9$()                                   : REM INIT FOR LINE FEED
2224  $GIO LF      #5 (A000         ,R8$(9))X9$()<1,7>     : REM ADJUST PAPER
2228  SELECT       PRINT 405(80)                           : REM SEL CONSOLE DEVICE
2232 REM **
2236  GOSUB ' 122                                          : REM DEVICE RESET
```

```
2240 REM **
2244 REM **                                                  INSTRUCITON FORMATS
2248 REM **
2252  % * * * * *  INSERT PAPER INTO TYPEWRITER  * * * *            #
2256  % * * * * *  SET LEFT MARGIN TO LEFT EDGE OF PAPER  * * * *  #
2260  % * * * * *  SET SPACING SELECTOR TO SINGLE SPACING  * * * * #
2264  %          * * * *  TYPEWRITER IS NOT READY  * * * *         #
2268  % * * * * *  FOLLOW INSTRUCTIONS ON OUTPUT DEVICE  * * * * * #
2272  %   Touch 'RETURN' When Completed >                           #
2276  %    #     AFTER EACH PAGE IS PRINTED THE PROGRAM WILL STOP.
2280  %    #     CHECK THAT THE PAPER IS PROPERLY ALIGNED...
2284  %    #     MAKE ADJUSTMENTS WHEN NESSARY.
2288  %    #     TO CONTINUE ... Switch the  'MANL/AUTO'  SWITCH TO
2292  %    #       'MANL'  POSITION THEN SWITCH TO  'AUTO' POSITION.
2296  %    # -=!----- ALIGN TOP EDGE OF PAPER TO THIS POSITION ...
2300  %    # THEN CONTINUE ------------!=-
2304 REM **
2308 REM **
2312  RETURN                                                 : REM EXIT
```

AUTHOR System: Phase 1 - ATH1/I - # MAP

0768 - 1108
0804 - 0804
0808 - 0956 0972 1000 1024
0820 - 0812 0828 0884
0836 - 0820
0860 - 0820
0880 - 0860 0868 0920
0900 - 0872 0876
0944 - 0932
0952 - 0944
0972 - 0900
1000 - 0988
1016 - 0900
1040 - 0900
1056 - 1048
1068 - 1040
1076 - 1060
1080 - 0852
1100 - 1092
1112 - 1084
1116 - 1080
1232 - 1212
1244 - 1256
1252 - 1244
1260 - 1256
1276 - 1212
1324 - 1312
1328 - 1320
1352 - 1212
1392 - 1212
1396 - 1384 1388
1408 - 1260 1336 1396
1532 - 1520
1544 - 1536
1568 - 1564
1572 - 1569
1580 - 1568
1612 - 1571
1616 - 1604
1644 - 1567 1612
1692 - 1680
1716 - 1680
1728 - 1680
1740 - 1708 1720 1732
1828 - 1828
1832 - 1828
1840 - 1784 1788 1832
1856 - 1808
1860 - 1812
1864 - 1824
2136 - 2136
2144 - 2136 2168
2172 - 2148
2236 - 2092
2252 - 2108
2256 - 2116
2260 - 2124
2264 - 2156

```
2268  - 2176
2272  - 2132
2276  - 2192
2280  - 2196
2284  - 2200
2288  - 2204
2292  - 2208
2296  - 2212
2300  - 2216
```

```
Q2$(   - 1524 1692 1700 1944
R0$(   - 0804 0820 0860 0864 0868 0872 0876 0900 0924 0948 0952 1256 1828 1832
         2136
R1     - 0764 0772 0780 0920 1076 1088 1092 1096
R1$    - 0764 0812 0944 1048 1080
R2     - 1088 1096 1100 1112
R2$    - 0796 0808 0840 0848 0924 0984 1016 1020 1044 1068 1072 1084 1112
R2$(   - 1508 1512 1544 1548
R3     - 0792 0844 0848 0920 0924 0928 0932 0972 0980 0984 1040 1068 1076 1092
R4     - 1580 1588 1596 1616 1624 1632 1644 1652 1660
R4$(   - 0864
R5     - 1584 1588 1592 1620 1624 1628 1648 1652 1656
R5$(   - 1532
R6$    - 0764 1044 1052 1072 1076
R8$    - 1928 1940 2004 2020
R8$(   - 0772 0780 0784 0808 0816 0824 0880 1100 1104 1932 1940 1944 2012 2020
         2024 2032 2144 2160 2224
R9     - 0788 0808 0936 0988 0992 1016 1020
R9$(   - 0768 0772 0776 0780 1100 2008 2032
W2$    - 1244 1784 1788 1792 1836
W2$(   - 1476 1516 1524 1532 1540 1544 1548 1692 1696 1700 1704 1716 1728 2024
W3$    - 1312
W3$(   - 1476 1532 1552 1652 1664
W4$    - 1384 1388
W5$    - 2092 2096
W6     - 1188 1212 1520 1536 1680
W6$    - 1564 1568 1569 1604 1612
W7     - 1292
W8     - 1304 1328 1368 1376
W9     - 1504 1516 1528 1532 1540 1544 1548 1696 1716 1728 1936 1940 1944 1948
         2016 2020 2024 2028
W9$(   - 1476 1484 1488 1492 1496 1500 1516 1540 1696 1716 1728
X3     - 1296
X3$(   - 1480 1544 1556 1565 1570 1588 1600
X6     - 1600 1636 1664
X7     - 1600 1636 1664
X9$(   - 2088 2220 2224
Y3     - 1300
Y3$(   - 1480 1548 1560 1566 1572 1624 1636
```

```
AUTHOR System: Phase 1 - ATH1/I    - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 101
       - 1800 1820 2104 2112 2120 2128 2152 2164
0764 DEFFN' 104
1188 DEFFN' 108
1472 DEFFN' 109
       - 1236 1280
1780 DEFFN' 110
       - 1240
1924 DEFFN' 111
       - 1248
2000 DEFFN' 112
       - 1252
2084 DEFFN' 113
       - 1284
???? DEFFN' 114
       - 1288 1332
???? DEFFN' 115
       - 1308
???? DEFFN' 116
       - 1316
???? DEFFN' 117
       - 1324 1372
???? DEFFN' 118
       - 1360
???? DEFFN' 119
       - 1364 1380
???? DEFFN' 120
       - 1392
???? DEFFN' 121
       - 2184
???? DEFFN' 122
       - 2236
???? DEFFN' 123
       - 1192
???? DEFFN' 124
       - 1196
???? DEFFN' 125
       - 1588 1624 1652
```

3.2.2 ATH2/I

This section of the 'AUTHOR/1' program contains subroutines 114 to 123.

The remainder of this section contains the 'ATH2/I' program listing and reference maps.

```
2500 %## Program > AUTHOR/1    (ATH2/I  )  Version > 2.0  2200 T/VP  780316
2504 REM **
2508 REM **
2512 REM **  Subroutine  ********************************************** 1 1 4
2516 REM **  Abstract >
2520 REM **    To print upper/lower hash lines on the TTY device.
2524 REM **  Variables >
2528 REM **    W7$() > hash control buffer  ! W9 > upper/lower index
2532 REM **
2536 REM ********************************************
2540  DEFFN ' 114 (W9)                                    : REM ENTRY
2544  DIM          W7$(19)4,W8$(1)64
2548       W7$(1)=HEX(09210921) : W7$(2)=HEX(08080808): REM INIT TABLE
2552       W7$(3)=HEX(5F5F5F5F) : W7$(4)=HEX(0A0A0A0A): REM INIT TABLE
2556       W7$(5)=HEX(0A0A0A0A) : W7$(6)=HEX(0A0A0A0A): REM INIT TABLE
2560       W7$(7)=HEX(0A0A0A0A) : W7$(8)=HEX(0A0A0A0A): REM INIT TABLE
2564       W8$(1)="AUTHOR: An Automatic Authoring System      Version:  2.0"
2568 REM **
2572 REM **       !   upper   !   lower   !
2576 REM **       !-----------!-----------!
2580  ON W9 GOTO      2592    ,   2636                   : REM BRANCH
2584 REM **       !-----------!-----------!
2588 REM **
2592  $CIO TAB    #5 (4009A020A000,R8$(9))W7$()<33,4> : REM HORZ TAB
2596  $CIO HEAD   #5 (A000        ,R8$(9))W8$()<1,56> : REM PRINT HEADING
2600  $CIO TUP    #5 (400D400A    ,R8$(9))            : REM VERTICAL TAB
2604  $CIO LINE 1 #5 (A000400D    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2608  $CIO LINE 2 #5 (A000400D    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2612  $CIO LINE 2 #5 (A000        ,R8$(9))W7$()<3,10> : REM PRINT HASH
2616  $CIO LINE 3 #5 (A000        ,R8$(9))W7$()<1,2>  : REM PRINT HASH
2620  $CIO LINE 3 #5 (A000400D    ,R8$(9))W7$()<8,5>  : REM PRINT HASH
2624  $CIO LF     #5 (A000        ,R8$(9))W7$()<13,3> : REM VERTICAL TAB
2628  GOTO         2668                               : REM BRANCH TO EXIT
2632 REM **
2636  $CIO LF     #5 (A000        ,R8$(9))W7$()<13,2> : REM VERTICAL TAB
2640  $CIO LINE 1 #5 (4009A000    ,R8$(9))W7$()<5,8>  : REM PRINT HASH
2644  $CIO LINE 1 #5 (4009A000400D,R8$(9))W7$()<9,4>  : REM PRINT HASH
2648  $CIO LINE 1 #5 (A000400D    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2652  $CIO LINE 2 #5 (A000400D    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2656  $CIO LINE 3 #5 (A000400D    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2660  $CIO LF     #5 (A000        ,R8$(9))W7$()<13,19>: REM VERTICAL TAB
2664 REM **
2668  RETURN                                           : REM EXIT
```

```
2672 REM **
2676 REM **
2680 REM **  Subroutine  ********************************************** 1 1 5
2684 REM **  Abstract >
2688 REM **    To scan lookup table for next underscored character.
2692 REM **  Variables >
2696 REM **    W7 > lookup table indx     ! W3$ > return scan flag
2700 REM **    W8 > text buffer lin indx  ! R2$() > search return buffer
2704 REM **    W1 > next char position    ! W3$() > lookup table
2708 REM **
2712 REM ***********************************************
2716  DEFFN ' 115                                          : REM ENTRY
2720  MAT COPY     W3$()<W7,2> TO R2$()<1,2>               : REM STOR BIN POSITION
2724               W1 = FNR(VAL(R2$(1)))                   : REM CMPUT BIN POSITION
2728  MAT COPY     X3$()<X3,2> TO R2$()<1,2>               : REM STOR BIN POSITION
2732               X4 = FNR(VAL(R2$(1)))                   : REM CMPUT BIN POSITION
2736  MAT COPY     Y3$()<Y3,2> TO R2$()<1,2>               : REM STOR BIN POSITION
2740               Y4 = FNR(VAL(R2$(1)))                   : REM CMPUT BIN POSITION
2744  REM **
2748  IF           W1 <= W8*64              THEN 2776      : REM SET POSITV SCAN
2752  IF           X4 <= W8*64              THEN 2776      : REM SET POSITV SCAN
2756  IF           Y4 <= W8*64              THEN 2776      : REM SET POSITV SCAN
2760  REM **
2764               W3$ = "NULL SCAN"                       : REM SET RETURN FLAG
2768  GOTO         2780                                    : REM BRANCH TO EXIT
2772 REM **
2776               W3$ = "POSITIVE SCAN"                   : REM SET RETURN FLAG
2780  RETURN                                               : REM EXIT
```

```
2784 REM **
2788 REM **
2792 REM **  Subroutine  ********************************************** 1 1 6
2796 REM **  Abstract >
2800 REM **    To print line containing underscoring on TTY device.
2804 REM **  Variables >
2808 REM **    W9$() > micro command buffr  ! W9 > temp text bufr position
2812 REM **    W2$() > text buffer          ! W7 > lookup table indx
2816 REM **    W3$() > lookup table         ! W2 > next character position
2820 REM **    R2$() > search return bufr   ! W1 > curnt char position
2824 REM **    R8$() > micro command buffer ! W8 > text buffr lin indx
2828 REM **  Function Definitions >
2832  DEFFN    T(T) = T - (W9+WO-1)         : REM compute string length
2836 REM **
2840 REM ********************************************
2844  DEFFN ' 116                                      : REM ENTRY
2848  DIM          W9$(5)8                             : REM DEFINE WORK BUFR
2852               W9$(1) = HEX(40094020202020)        : REM INIT MICRO CMND
2856               W9$(2) = HEX(A0004008405F2020)      : REM INIT MICRO CMND
2860               W9$(3) = HEX(A000400D20202020)      : REM INIT MICRO CMND
2864               W9$(4) = HEX(A0004008402D2020)      : REM INIT MICRO CMND
2868               W9$(5) = HEX(A0004008402F2020)      : REM INIT MICRO CMND
2872 REM **
2876               WO     = 0                          : REM SET TEMP LIN INDX
2880               W9     = (W8-1)*64 + 1              : REM CMPUT BEG OF LINE
2884  $GIO TAB     #5 (W9$(1),R8$(9))                  : REM HORZ TAB
2888  REM **
2892  IF           W1 > X4                   THEN 2952 : REM W1 NOT MIN VALUE
2896  IF           W1 > Y4                   THEN 2972 : REM W1 NOT MIN VALUE
2900  $GIO PRINT   #5 (W9$(2),R8$(9))W2$()<W9+WO,FNT(W1)>: REM PRINT STRING
2904  IF           W1 <> X4                  THEN 2928 : REM ALSO SPCL CHAR 1
2908  $GIO PRINT   #5 (W9$(4),R8$(9))W2$()<W9+WO,0>    : REM PRINT SPCL CHAR 1
2912  GOSUB 3144                                       : REM INCR TBL INDX X3
2916  GOSUB 3076                                       : REM INCR TBL INDX W3
2920  GOTO         2988                                : REM BRNCH TO LIMIT CHK
2924  REM **
2928  IF           W1 <> Y4                  THEN 2940 : REM ALSO SPCL CHAR 2
2932  $GIO PRINT   #5 (W9$(5),R8$(9))W2$()<W9+WO,0>    : REM PRINT SPCL CHAR 2
2936  GOSUB 3212                                       :  REM INCR TBL INDX Y3
2940  GOSUB 3076                                       :  REM INCR TBL INDX W3
2944  GOTO         2988                                : REM BRNCH TO LIMIT CHK
2948  REM **
2952  IF           X4 > Y4                   THEN 2972 : REM X4 NOT MIN VALUE
2956  $GIO PRINT   #5 (W9$(4),R8$(9))W2$()<W9+WO,FNT(X4)>: REM PRINT STRING
2960  GOSUB 3144                                       : REM INCR TBL INDX X3
2964  GOTO  2988                                       : REM BRNCH TO LIMIT CHK
```

```
2968  REM **
2972  $GIO PRINT  #5 (W9$(5),R8$(9))W2$()<W9+W0,FNT(Y4)>: REM PRINT STRING
2976  GOSUB 3212                                       : REM INCR TBL INDX Y3
2980  GOTO  2988                                       : REM BRNCH TO LIMIT CHK
2984  REM **
2988  IF          W1 <= W8*64                THEN 2892 : REM CONT THIS LINE
2992  IF          X4 <= W8*64                THEN 2892 : REM CONT THIS LINE
2996  IF          Y4 <= W8*64                THEN 2892 : REM CONT THIS LINE
3000  REM **
3004  MAT COPY    -W2$()<W9,64> TO R9$()<1,64>         : REM REVERSE LIN COPY
3008  MAT SEARCH  R9$(), <> R5$(1) TO R2$()            : REM SRCH FOR NON BLNK
3012              W2 = FNR(VAL(R2$(1)))                : REM CMPUT TRL SPCS
3016              W2 = (64-(W2-1)) - W0                : REM CMPUT TRL SPACES
3020  $GIO END    #5 (W9$(3),R8$(9))W2$()<W9+W0,W2>    : REM PRINT END OF LINE
3024  RETURN                                           : REM EXIT
```

```
3028 REM **
3032 REM **
3036 REM **  Subroutine  ********************************************** 1 1 6.1
3040 REM **  Abstract >
3044 REM **    This routine will increment and compute the index
3048 REM **    into the underscore lookup table.
3052 REM **  Variables >
3056 REM **    W7 > lookup table index  ! W3$() > lookup table
3060 REM **    W1 > compute position    ! WO > temp bufr lin indx
3064 REM **    W9 > curnt begin of line ! R2$() > function input bufr
3068 REM **
3072 REM ***********************************************
3076              W7 = W7 + 2                              : REM INCR LOKUP TBL IDX
3080  MAT COPY    W3$()<W7,2> TO R2$()<1,2>                : REM FTCH NXT CHAR POS
3084              WO = (W1+1) - W9                         : REM INCR TMP LIN INDX
3088              W1 = FNR(VAL(R2$(1)))                    : REM CMPUT DEC CHAR POS
3092  RETURN                                               : REM EXIT
```

```
3096 REM **
3100 REM **
3104 REM **  Subroutine  ********************************************** 1 1 6.2
3108 REM **  Abstract >
3112 REM **    This routine will increment and compute the index
3116 REM **    into the 'Z' scan lookup table.
3120 REM **  Variables >
3124 REM **    X3 > lookup table index  ! X3$() > lookup table
3128 REM **    X4 > compute position    ! WO > temp bufr lin indx
3132 REM **    W9 > curnt begin of line ! R2$() > function input bufr
3136 REM **
3140 REM ************************************************
3144                X3 = X3 + 2                          : REM INCR LOKUP TBL IDX
3148  MAT COPY      X3$()<X3,2> TO R2$()<1,2>            : REM FTCH NXT CHAR POS
3152                WO = (X4+1) - W9                     : REM INCR TMP LIN INDX
3156                X4 = FNR(VAL(R2$(1)))                : REM CMPUT DEC CHAR POS
3160  RETURN                                             : REM EXIT
```

```
3164 REM **
3168 REM **
3172 REM **  Subroutine  ********************************************** 1 1 6.3
3176 REM **  Abstract >
3180 REM **    This routine will increment and compute the index
3184 REM **    into the '0' Scan lookup table.
3188 REM **  Variables >
3192 REM **    Y3 > lookup table index  ! Y3$() > lookup table
3196 REM **    Y4 > compute position    ! W0 > temp bufr lin indx
3200 REM **    W9 > curnt begin of line ! R2$() > function input bufr
3204 REM **
3208 REM ***********************************************
3212              Y3 = Y3 + 2                              : REM INCR LOKUP TBL IDX
3216  MAT COPY    Y3$()<Y3,2> TO R2$()<1,2>                : REM FTCH NXT CHAR POS
3220              W0 = (Y4+1) - W9                         : REM INCR TMP LIN INDX
3224              Y4 = FNR(VAL(R2$(1)))                    : REM CMPUT DEC CHAR POS
3228  RETURN                                               : REM EXIT
```

```
3232 REM **
3236 REM **
3240 REM **  Subroutine  ********************************************** 1 1 7
3244 REM **  Abstract >
3248 REM **    To print a line without trailing blanks on selected
3252 REM **    output device.
3256 REM **  Variables >
3260 REM **    W9 > temp text bufr indx  ! W2$() > text buffer
3264 REM **    R9$() > temp line bufr    ! R5$() > single char storage
3268 REM **    R2$() > serach retrn bufr ! W6 > output device indx
3272 REM **    W0 > temp length of line  ! W7 > null line flag
3276 REM **
3280 REM ***********************************************
3284  DEFFN ' 117                                          : REM ENTRY
3288  DIM          W9$(5)8                                 : REM DEFINE WORK BUFR
3292               W9$(1) = HEX(5FA0)                      : REM DEFINE CONVERSN
3296               W9 = (W8-1)*64 + 1                      : REM CMPUT BEG OF LIN
3300  $TRAN        (W2$()<W9,64>, W9$()<1,8>) R            : REM MINIZE UNDERSCOR
3304  MAT COPY     -W2$()<W9,64> TO R9$()<1,64>            : REM REVERSE COPY
3308  MAT SEARCH   R9$(), <> R5$(1) TO R2$()               : REM SCAN FOR NONBLANKS
3312  IF           R2$(1) <> HEX(0000)       THEN 3324     : REM TRAILING BLANKS
3316               BIN(STR(R2$(1),1,1)) = 0                : REM SET TO NULL LINE
3320               BIN(STR(R2$(1),2,1)) = 65               : REM SET TO NULL LINE
3324               W0 = 64 - (FNR(VAL(R2$(1)))-1)          : REM CMPUT LEN OF STR
3328               BIN(STR(R8$(9),6,1)) = W0               : REM STOR LEN OF STR
3332               BIN(STR(R8$(9),7,1)) = 00               : REM SET LIMIT
3336  $GIO SETUP   #5 (R8$(8),R8$(9))                      : REM SETUP CR OR LF
3340 REM **
3344 REM **        !   CRT   !   TTY   !   LPT   !
3348 REM **        !---------!---------!---------!
3352  ON W6 GOTO     3400   ,  3364   ,  3380              : REM BRANCH
3356 REM **        !---------!---------!---------!
3360 REM **
3364  $GIO TAB     #5 (14671608400940Z0,R8$(9))           : REM HORZ TAB
3368  $GIO TEXT    #5 (A0004250     ,R8$(9))W2$()<W9,W0>: REM PRINT LINE
3372  GOTO         3400                                   : REM BRANCH TO EXIT
3376 REM **
3380  INIT(20)     R9$()                                  : REM CLR TAB BUFFER
3384  $GIO TAB     #5 (14671608A000,R8$(9))R9$()<1,34>    : REM HORZ TAB
3388  $GIO TEXT    #5 (A0004250     ,R8$(9))W2$()<W9,W0>: REM PRINT LINE
3392  GOTO         3400                                   : REM BRANCH TO EXIT
3396 REM **
3400  RETURN                                              : REM EXIT
```

```
3404 REM **
3408 REM **
3412 REM **  Subroutine  ********************************************** 1 1 8
3416 REM **  Abstract >
3420 REM **    To Display Instructions to set up LPT device.
3424 REM **  Variables >
3428 REM **    W4$ > return scan flag      ! W4$() > i/o buffer #1
3432 REM **    W9  > scratch loop indx     ! W5$() > i/o buffer #2
3436 REM **    R0$() > input register      ! W6$ > buffer scan flag
3440 REM **
3444 REM ***********************************************
3448  DEFFN ' 118                                          : REM ENTRY
3452  DIM          X9$(10)45                               : REM DEFINE WORK BUFRS
3456  IF           W4$ = "NO UNDERSCORE"      THEN 4008    : REM SKIP ROUTINE
3460  IF           W4$ = "UNDERSCORE"         THEN 4008    : REM SKIP ROUTINE
3464  IF           W4$ = "OVERLAY"            THEN 4008    : REM SKIP ROUTINE
3468  IF           W4$ = "TAPE"               THEN 4008    : REM SKIP ROUTINE
3472  IF           W4$ = "SET PRINTER"        THEN 3532    : REM INIT PRINTER ONLY
3476               W4$ = "NO UNDERSCORE"                   : REM SET RETRN SCAN FLG
3480  PRINT        HEX(03)                                 : REM CLR CRT SCREEN
3484  GOSUB ' 101  (5,1,0,0)                               : REM POSITION CUSOR
3488  SELECT       PRINT 005                               : REM SEL OUTPUT MODE
3492  PRINTUSING   3824, " "                               : REM PRINT INSTRUTN 1
3496  PRINTUSING   3828, " "                               : REM PRINT INSTRUTN 2
3500  PRINTUSING   3832, " "                               : REM PRINT INSTRUTN 3
3504  SELECT       PRINT 405                               : REM SEL OUTPUT MODE
3508  GOSUB ' 101  (2,1,0,0)                               : REM POSITION CUSOR
3512  PRINTUSING   3836, " ";                              : REM PRINT PROMPT
3516  KEYIN        R0$(1), 3520, 3520       : GOTO 3516    : REM RESPONCE WAIT
3520  IF           R0$(1) <> "1"              THEN 3528    : REM SCAN FLAG SET
3524               W4$ = "UNDERSCORE"                      : REM SET SCAN FLAG
3528  IF           R0$(1) <> "2"              THEN 3544    : REM SCAN FLAG SET
3532               W4$ = "TAPE"                            : REM SET SCAN FLAG
3536  GOSUB ' 190  (9,2,Z3,0,1,6)                          : REM REWIND TAPE
3540 REM **
3544  $GIO READY   #5 (4100          ,R8$(9))              : REM SEND CTL CHAR
3548  $IF  ON      #5, 3572                                : REM DEVICE READY
3552  GOSUB ' 101  (11,10,0,0)                             : REM POSITION CUSOR
3556  PRINTUSING   3840, " "                               : REM PRINT ERR MSG
3560  $GIO PAUSE   /005(R8$(4),R8$(9))                     : REM ERROR TONE
3564  GOSUB ' 101  (11,10,1,0)                             : REM ERASE ERR MSG
3568  GOTO         3544                                    : REM CONT READY TEST
3572  PRINT        HEX(030A0A0A0A0A0A)                     : REM CLR CONSOLE DEV
3576  PRINTUSING   3844, " "                               : REM PRINT INSTRUCTN
```

```
3580 REM **
3584  GOSUB 3856                                          : REM INIT TEXT BUFFR 1
3588  GOSUB ' 126 ( 33,13)                                : REM VERT TAB
3592  GOSUB ' 126 ( 46,30)                                : REM LEFT ARROWS
3596  GOSUB ' 126 (361, 5)                                : REM HORZ TAB
3600  GOSUB ' 126 (136,30)                                : REM TEXT
3604  GOSUB ' 126 ( 91,30)                                : REM RIGHT ARROWS
3608  GOSUB ' 126 ( 91,35)                                : REM RIGHT ARROWS
3612  GOSUB ' 126 ( 44, 2)                                : REM CRLF
3616  REM **
3620  GOSUB ' 126 (181,30)                                : REM LEFT MARGIN
3624  GOSUB ' 126 (226,40)                                : REM SIZE
3628  GOSUB ' 126 (361,35)                                : REM HORZ TAB
3632  GOSUB ' 126 (316,25)                                : REM RIGHT MARGIN
3636  GOSUB ' 126 ( 44, 2)                                : REM CRLF
3640  REM **
3644  GOSUB ' 126 (361, 0)                                : REM HORZ TAB
3648  GOSUB ' 126 (181,30)                                : REM LEFT MARGIN
3652  GOSUB ' 126 (271,20)                                : REM SIZE
3656  GOSUB ' 126 (316,25)                                : REM RIGHT MARGIN
3660  GOSUB ' 126 ( 44, 2)                                : REM CRLF
3664  REM **
3668  REM **
3672  GOSUB 3908                                          : REM INIT TEXT BUFFR 2
3676  GOSUB ' 126 (285,10)                                : REM HORZ TAB
3680  GOSUB ' 126 (406,40)                                : REM HORZ LINE
3684  GOSUB ' 126 (406,40)                                : REM HORZ LINE
3688  GOSUB ' 126 ( 44, 2)                                : REM CRLF
3692  REM **
3696  FOR         X7 = 2 TO 9                             : REM BEG LINE LOOP
3700  GOSUB ' 126 (285,30)                                : REM HORZ TAB
3704  GOSUB ' 126 (((X7-1)*45)+1,LEN(X9$(X7)))            : REM TEXT
3708  GOSUB ' 126 ( 44, 2)                                : REM CRLF
3712  NEXT        X7                                      : REM CONT LINE LOOP
3716  REM **
3720  GOSUB ' 126 (285,10)                                : REM HORZ TAB
3724  GOSUB ' 126 (406,40)                                : REM HORZ LINE
3728  GOSUB ' 126 (406,40)                                : REM HORZ LINE
3732  GOSUB ' 126 ( 44, 2)                                : REM CRLF
3736  REM **
3740  REM **
3744  GOSUB 3968                                          : REM INIT TEXT BUFFR 3
3748  FOR         X7 = 2 TO 8                             : REM BEGIN LINE LOOP
3752  GOSUB ' 126 (361,30)                                : REM HORZ TAB
3756  GOSUB ' 126 (((X7-1)*45)+1,LEN(X9$(X7)))            : REM TEXT
3760  GOSUB ' 126 ( 44, 2)                                : REM CRLF
3764  NEXT        X7                                      : REM CONT LINE LOOP
3768  REM **
3772  REM **
3776  $GIO TOP    #5 (400C     ,R8$(9))                   : REM ISSUE FORM FEED
3780  KEYIN       RO$(1), 3800, 3780                      : REM RESPONCE WAIT
3784  IF          RO$(1) = HEX(81)          THEN 3780     : REM NO TONE
3788  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
3792  GOTO        3780                                    : REM CONTINUE WAIT
3796  REM **
3800  IF          RO$(1) = HEX(84)          THEN 3584     : REM REPRINT PAGE
3804  IF          RO$(1) = HEX(81)          THEN 3780     : REM SHUT OFF TONE
3808  GOTO        4008                                    : REM BRANCH TO EXIT
```

```
3812 REM **
3816 REM **                                                  INSTRUCITON FORMATS
3820 REM **
3824  % LPT UNDERSCORE MENU #  1 -  SINGLE PAGE UNDERSCORING
3828  %                      #  2 -  MULTI PAGE UNDERSCORING (W/ TAPE)
3832  %                      #  3 -  DO NOT SCAN FOR UNDERSCORING
3836  % # Enter Code from LPT Underscoring Menu >
3840  %         * * * *  LINEPRINTER IS NOT READY  * * * *       #
3844  % * * * *  FOLLOW INSTRUCTIONS ON OUTPUT DEVICE  * * * *   #
3848  REM **
3852  REM **
3856  INIT (20)    X9$()                                        : REM LOAD TEXT BUFR 1
3860  INIT (0A)    X9$(01)
3864  INIT (0D)    STR(X9$(1),45,1)
3868  INIT ("<")   X9$(02)
3872  INIT (">")   X9$(03)
3876               X9$(04) = "Align to Sixth Paper Hole"
3880               X9$(05) = "<<<<<------- Left Margin"
3884               X9$(06) = "14 7/8  X  11"
3888               X9$(07) = "08 1/2  X  11"
3892               X9$(08) = "Right Margin ------->>>>>"
3896  RETURN                                                  : REM EXIT
3900  REM **
3904  REM **
3908  INIT (20)    X9$()                                        : REM LOAD TEXT BUFR 2
3912  INIT (0A)    X9$(1)
3916  INIT (0D)    STR(X9$(1),45,1)
3920  INIT (2D)    X9$(10)
3924               X9$(2)  = "This page should now be aligned"
3928               X9$(3)  = "to the desired position..."
3932               X9$(4)  = "--  To Continue to print THIS PAGE"
3936               X9$(5)  = "    TOUCH the 'CONTINUE' Key."
3940               X9$(6)  = "--  To Begin Listing TOUCH the"
3944               X9$(7)  = "    'RUN' Key."
3948               X9$(8)  = "--  This page should appear on"
3952               X9$(9)  = "    an UPWARD Facing Page."
3956  RETURN                                                  : REM EXIT
3960  REM **
3964  REM **
3968  INIT (20)    X9$()                                        : REM LOAD TEXT BUFR 3
3972  INIT (0A)    X9$(1)
3976  INIT (0D)    STR(X9$(1),45,1)
3980               X9$(2)  = "Special Instructions for Underscoring:"
3984               X9$(3)  = "--  Single Page Overlaying:  At the"
3988               X9$(4)  = "    sound of the tone, position page"
3992               X9$(5)  = "    two inches above top of form."
3996               X9$(6)  = "--  Multi Page Overlaying:  At the"
4000               X9$(7)  = "    sound of the tone, position paper"
4004               X9$(8)  = "    to this page and position."
4008  RETURN                                                  : REM EXIT
```

```
4012 REM **
4016 REM **
4020 REM **  Subroutine  ********************************************** 1 1 9
4024 REM **  Abstrct >
4028 REM **    To print upper/lower hash lines on the LPT device.
4032 REM **  Variables >
4036 REM **    W7$() > hash contorl buffer  ! W9 > upper/lower index
4040 REM **
4044 REM ********************************************
4048  DEFFN ' 119 (W9)                                        : REM ENTRY
4052  DIM          W7$(19)4,W8$(1)64
4056  INIT(20)     W7$()                                      : REM CLR TABLE
4060       W7$(1)=HEX(2D2D2D2D) : W7$(2)=HEX(0A0A0A0A): REM INIT TABLE
4064       W8$(1)="AUTHOR: An Automatic Authoring System      Version:  2.0"
4068 REM **
4072 REM **        !   upper    !   lower    !
4076 REM **        !------------!------------!
4080  ON W9 GOTO       4092     ,   4152                    : REM BRANCH
4084 REM **        !------------!------------!
4088 REM **
4092  $GIO TOP     #5 (400CA000     ,R8$(9))W7$()<5,3>  : REM TOP OF FORM
4096  $GIO TAB     #5 (A000         ,R8$(9))W7$()<9,38> : REM HORZ TAB
4100  $GIO HEAD    #5 (A000400D     ,R8$(9))W8$()<1,64> : REM PRINT HEADING
4104  $GIO LF      #5 (A000         ,R8$(9))W7$()<5,3>  : REM VERTICAL TAB
4108  FOR          W9 = 1 TO 3                          : REM PRINT 3 LINES
4112  $GIO TAB     #5 (A000407C     ,R8$(9))W7$()<9,33> : REM HORZ TAB/HASH
4116  $GIO TAB     #5 (A000407C400D,R8$(9))W7$()<9,64> : REM HORZ TAB/HASH
4120  NEXT         W9                                   : REM PRINT NXT LINE
4124  $GIO TAB     #5 (A000         ,R8$(9))W7$()<9,30> : REM HORZ TAB
4128  $GIO HASH    #5 (A000         ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4132  $GIO TAB     #5 (A000         ,R8$(9))W7$()<9,64> : REM HORZ TAB
4136  $GIO HASH    #5 (A000400D     ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4140  $GIO LF      #5 (A000         ,R8$(9))W7$()<5,3>  : REM VERTICAL TAB
4144  GOTO         4192                                 : REM BRANCH TO EXIT
4148 REM **
4152  $GIO LF      #5 (A000         ,R8$(9))W7$()<5,3>  : REM VERTICAL TAB
4156  $GIO TAB     #5 (A000         ,R8$(9))W7$()<9,30> : REM HORZ TAB
4160  $GIO HASH    #5 (A000         ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4164  $GIO TAB     #5 (A000         ,R8$(9))W7$()<9,64> : REM HORZ TAB
4168  $GIO HASH    #5 (A000400D     ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4172  FOR          W9 = 1 TO 3                          : REM PRINT 3 LINES
4176  $GIO TAB     #5 (A000407C     ,R8$(9))W7$()<9,33> : REM HORZ TAB/HASH
4180  $GIO TAB     #5 (A000407C400D,R8$(9))W7$()<9,64> : REM HORZ TAB/HASH
4184  NEXT         W9                                   : REM PRINT NXT LINE
4188 REM **
4192  RETURN                                            : REM EXIT
```

```
4196 REM **
4200 REM **
4204 REM **  Subroutine  ********************************************** 1 2 0
4208 REM **  Abstract >
4212 REM **    To print the underscore overlay buffer on to LPT device.
4216 REM **  Variables >
4220 REM **    W6 > output device indx    ! W4$ > return scan flag
4224 REM **    Z3 > end of file flag      !
4228 REM **
4232 REM ***********************************************
4236  DEFFN ' 120                                          : REM ENTRY
4240  IF          W6 = 4                    THEN 4292    : REM END OF PRNT FILE
4244  GOSUB ' 109                                        : REM CREATE OVERLAY
4248  IF          W4$ = "TAPE"              THEN 4280    : REM WRITE ON TAPE
4252 REM **
4256  GOSUB ' 122                                        : REM DEVICE WAIT
4260              W4$ = "OVERLAY"                        : REM SET SCAN FLAG
4264  GOSUB ' 108 (W6)                                   : REM PRINT BUFFR
4268              W4$ = "UNDERSCORE"                     : REM RESET SCAN FLAG
4272  GOTO        4344                                   : REM BRANCH TO EXIT
4276 REM **
4280  GOSUB ' 190 (12,1,Z3,2048,1,6)                     : REM WRITE BUFFER
4284  GOTO        4344                                   : REM BRANCH TO EXIT
4288 REM **
4292              W6 = 3                                 : REM RESET DEVICE FLG
4296  GOSUB ' 190 (10,3,Z3,0,1,6)                        : REM WRITE EOF
4300  DEFFN ' 016                                        : REM RECOVERY ENTRY
4304  GOSUB ' 190 (9,3,Z3,0,1,6)                         : REM REWIND TAPE
4308  GOSUB ' 122                                        : REM DEVICE WAIT
4312              W4$ = "OVERLAY"                        : REM SET SCAN FLAG
4316  GOSUB ' 190 (7,1,Z3,2048,1,6)                      : REM READ RECORD
4320  IF          Z3 = 4                    THEN 4332    : REM END OF FILE
4324  GOSUB ' 108 (3)                                    : REM PRINT BUFFER
4328  GOTO        4316                                   : REM READ NXT RECORD
4332  GOSUB ' 190 (9,2,Z3,0,1,6)                         : REM REWIND TAPE
4336              W4$ = "TAPE"                           : REM RESET SCAN FLAG
4340 REM **
4344  RETURN                                             : REM EXIT
```

```
4248  IF          W4$ = "TAPE"                THEN 4280 : REM WRITE ON TAPE
4252 REM **
4256  GOSUB ' 122                                       : REM DEVICE WAIT
4260              W4$ = "OVERLAY"                       : REM SET SCAN FLAG
4264  GOSUB ' 108 (W6)                                  : REM PRINT BUFFR
4268              W4$ = "UNDERSCORE"                    : REM RESET SCAN FLAG
4272  GOTO        4344                                  : REM BRANCH TO EXIT
4276 REM **
4280  GOSUB ' 190 (12,1,Z3,2048,1,6)                    : REM WRITE BUFFER
4284  GOTO        4344                                  : REM BRANCH TO EXIT
4288 REM **
4292              W6 = 3                                : REM RESET DEVICE FLG
4296  GOSUB ' 190 (10,3,Z3,0,1,6)                       : REM WRITE EOF
4300  DEFFN ' 016                                       : REM RECOVERY ENTRY
4304  GOSUB ' 190 (9,3,Z3,0,1,6)                        : REM REWIND TAPE
4308  GOSUB ' 122                                       : REM DEVICE WAIT
4312              W4$ = "OVERLAY"                       : REM SET SCAN FLAG
4316  GOSUB ' 190 (7,1,Z3,2048,1,6)                     : REM READ RECORD
4320  IF          Z3 = 4                      THEN 4332 : REM END OF FILE
4324  GOSUB ' 108 (3)                                   : REM PRINT BUFFER
4328  GOTO        4316                                  : REM READ NXT RECORD
4332  GOSUB ' 190 (9,2,Z3,0,1,6)                        : REM REWIND TAPE
4336              W4$ = "TAPE"                          : REM RESET SCAN FLAG
4340 REM **
4344  RETURN                                            : REM EXIT
4348 REM **
4352 REM **
4356 REM **  Subroutine  ********************************************** 1 2 1
4360 REM **  Abstract >
4364 REM **    To Clear all tab stops and set the needed tab stops.
4368 REM **  Variables >
4372 REM **    W9 > scratch loop indx     ! R9$() > temp tab buffer
4376 REM **    W8 > scratch loop indx     !
4380 REM **
4384 REM ***********************************************
4388  DEFFN ' 121                                       : REM ENTRY
4392 REM **
4396  FOR         W9 = 9 TO 1 STEP -4                   : REM DECRMNT LOOP
4400  $GIO CRLF   #5 (400D          ,R8$(9))            : REM CRLF
4404  FOR         W8 = 1 TO W9                          : REM TAB LOOP
4408  $GIO CLRTAB #5 (40094019      ,R8$(9))            : REM TAB/CLEAR
4412  NEXT        W8                                    : REM NEXT TAB
4416  NEXT        W9                                    : REM DECMNT LOOP
4420 REM **
4424  INIT(20)    R9$()                                 : REM CLR TAB BUFFR
4428  $GIO CRLF   #5 (400D          ,R8$(9))            : REM CRLF
4432  $GIO TABSET #5 (A000401A      ,R8$(9))R9$()<1,18> : REM TAB AND SET
4436  $GIO TABSET #5 (4020A000401A,R8$(9))R9$()<1,64> : REM TAB AND SET
4440  $GIO CRLF   #5 (400D          ,R8$(9))            : REM CRLF
4444 REM **
4448  RETURN                                            : REM EXIT
```

```
4452 REM **
4456 REM **
4460 REM **  Subroutine  ********************************************** 1 2 2
4464 REM **  Abstract >
4468 REM **    To allow the output device to be reset.  Program will
4472 REM **    pause until the device is disabled then will
4476 REM **    continue until a device ready is sensed.
4480 REM **  Variables >
4484 REM **    W9$() > micro command buffer  ! W6 > output device indx
4488 REM **
4492 REM ***********************************************
4496  DEFFN ' 122                                          : REM ENTRY
4500  DIM          W9$(5)8                                 : REM DEFINE WORK BUFR
4504               W9$(1) = HEX(122141002020)              : REM INIT MICRO CMND
4508               W9$(2) = HEX(122145002020)              : REM INIT MICRO CMND
4512               W9$(3) = HEX(122141002020)              : REM INIT MICRO CMND
4516  $GIO TONE    /005 (R8$(4),R8$(9))                    : REM ATTENTION TONE
4520  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
4524  $IF ON       #5,                              4516 : REM IF READY
4528  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
4532  $IF ON       #5,                              4540 : REM IF READY
4536                                               GOTO 4528 : REM IF NOT READY
4540  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
4544  $IF ON       #5,                              4552 : REM IF READY
4548                                               GOTO 4520 : REM IF NOT READY
4552  RETURN                                               : REM EXIT
```

```
4556 REM **
4560 REM **
4564 REM **  Subroutine  ********************************************** 1 2 3
4568 REM **  Abstract >
4572 REM **    To load 32 line text buffer for 16 line input buffer.
4576 REM **  Variables >
4580 REM **    Q2$() > input text buffer    ! W2$() > 32 line text buffer
4584 REM **    W9 > temp text bufr indx     ! W8 > temp text bufr indx
4588 REM **    W6 > output device indx      !
4592 REM **
4596 REM ***********************************************
4600  DEFFN ' 123                                          : REM ENTRY
4604 REM **
4608 REM **        !   CRT   !   TTY   !   LPT   !
4612 REM **        !---------!---------!---------!
4616  ON W6 GOTO     4628  ,  4644  ,  4640          : REM BRANCH
4620 REM **        !---------!---------!---------!
4624 REM **
4628  MAT COPY     Q2$()<1,1024> TO W2$()<1,2048>       : REM LOAD TEXT BUFR
4632  GOTO         4664                                 : REM BRANCH TO EXIT
4636 REM **
4640  IF           W4$ = "OVERLAY"          THEN 4664   : REM SKIP ADJUSTMENT
4644  FOR          W9 = 1 TO 1024 STEP 64               : REM BEG BUFR LOOP
4648               W8 = (W9*2) - 1                      : REM CMPUT NEW POS
4652  MAT COPY     Q2$()<W9,64> TO W2$()<W8,128>        : REM LOAD TEXT BUFR
4656  NEXT         W9                                   : REM CONT LOAD
4660 REM **
4664  RETURN                                            : REM EXIT
```

AUTHOR System: Phase 1 - ATH2/I - # MAP

```
2592 - 2580
2636 - 2580
2668 - 2628
2776 - 2748 2752 2756
2780 - 2768
2892 - 2988 2992 2996
2928 - 2904
2940 - 2928
2952 - 2892
2972 - 2896 2952
2988 - 2920 2944 2964 2980
3076 - 2916 2940
3144 - 2912 2960
3212 - 2936 2976
3324 - 3312
3364 - 3352
3380 - 3352
3400 - 3352 3372 3392
3516 - 3516
3520 - 3516
3528 - 3520
3532 - 3472
3544 - 3528 3568
3572 - 3548
3584 - 3800
3780 - 3780 3784 3792 3804
3800 - 3780
3824 - 3492
3828 - 3496
3832 - 3500
3836 - 3512
3840 - 3556
3844 - 3576
3856 - 3584
3908 - 3672
3968 - 3744
4008 - 3456 3460 3464 3468 3808
4092 - 4080
4152 - 4080
4192 - 4144
4280 - 4248
4292 - 4240
4316 - 4328
4332 - 4320
4344 - 4272 4284
4516 - 4524
4520 - 4548
4528 - 4536
4540 - 4532
4552 - 4544
4628 - 4616
4640 - 4616
4644 - 4616
4604 - 4632 4640
```

```
Q2$(   - 4628 4652
RO$(   - 3516 3520 3528 3780 3784 3800 3804
R2$(   - 2720 2724 2728 2732 2736 2740 3008 3012 3080 3088 3148 3156 3216 3224
         3308 3312 3316 3320 3324
R5$(   - 3008 3308
R8$(   - 2592 2596 2600 2604 2608 2612 2616 2620 2624 2636 2640 2644 2648 2652
         2656 2660 2884 2900 2908 2932 2956 2972 3020 3328 3332 3336 3364 3368
         3384 3388 3644 3560 3776 3788 4092 4096 4100 4104 4112 4116 4124 4128
         4132 4136 4140 4152 4156 4160 4164 4168 4176 4180 4400 4408 4428 4432
         4436 4440 4516 4520 4528 4540
R9$(   - 3004 3008 3304 3308 3380 3384 4424 4432 4436
T      - 2832
WO     - 2832 2876 2900 2908 2932 2956 2972 3016 3020 3084 3152 3220 3324 3328
         3368 3388
W1     - 2724 2748 2892 2896 2900 2904 2928 2988 3084 3088
W2     - 3012 3016 3020
W2$(   - 2900 2908 2932 2956 2972 3004 3020 3300 3304 3368 3388 4628 4652
W3$    - 2764 2776
W3$(   - 2720 3080
W4$    - 3456 3460 3464 3468 3472 3476 3524 3532 4248 4260 4268 4312 4336 4640
W6     - 3352 4240 4264 4292 4520 4528 4540 4616
W7     - 2720 3076 3080
W7$(   - 2544 2548 2552 2556 2560 2592 2604 2608 2612 2616 2620 2624 2636 2640
         2644 2648 2652 2656 2660 4052 4056 4060 4092 4096 4104 4112 4116 4124
         4128 4132 4136 4140 4152 4156 4160 4164 4168 4176 4180
W8     - 2748 2752 2756 2880 2988 2992 2996 3296 4404 4412 4648 4652
W8$(   - 2544 2564 2596 4052 4064 4100
W9     - 2540 2580 2832 2880 2900 2908 2932 2956 2972 3004 3020 3084 3152 3220
         3296 3300 3304 3368 3388 4048 4080 4108 4120 4172 4184 4396 4404 4416
         4644 4648 4652 4656
W9$(   - 2848 2852 2856 2860 2864 2868 2884 2900 2908 2932 2956 2972 3020 3288
         3292 3300 4500 4504 4508 4512 4520 4528 4540
X3     - 2728 3144 3148
X3$(   - 2728 3148
X4     - 2732 2752 2892 2904 2952 2956 2992 3152 3156
X7     - 3696 3704 3712 3748 3756 3764
X9$(   - 3452 3704 3756 3856 3860 3864 3868 3872 3876 3880 3884 3888 3892 3908
         3912 3916 3920 3924 3928 3932 3936 3940 3944 3948 3952 3968 3972 3976
         3980 3984 3988 3992 3996 4000 4004
Y3     - 2736 3212 3216
Y3$(   - 2736 3216
Y4     - 2740 2756 2896 2928 2952 2972 2996 3220 3224
Z3     - 3536 4280 4296 4304 4316 4320 4332
```

```
AUTHOR System:  Phase 1  -  ATH2/I     -  ' MAP

4300 DEFFN' 16
???? DEFFN' 101
       - 3484 3508 3552 3564
???? DEFFN' 108
       - 4264 4324
???? DEFFN' 109
       - 4244
2540 DEFFN' 114
2716 DEFFN' 115
2844 DEFFN' 116
3284 DEFFN' 117
3448 DEFFN' 118
4048 DEFFN' 119
4236 DEFFN' 120
4388 DEFFN' 121
4496 DEFFN' 122
       - 4256 4308
4600 DEFFN' 123
???? DEFFN' 126
       - 3588 3592 3596 3600 3604 3608 3612 3620 3624 3628 3632 3636 3644 3648
         3652 3656 3660 3676 3680 3684 3688 3700 3704 3708 3720 3724 3728 3732
         3752 3756 3760
???? DEFFN' 190
       - 3536 4280 4296 4304 4316 4332
```

3.2.3 ATH3/I

This section of the 'AUTHOR/1' program contains subroutines 124 to 126.

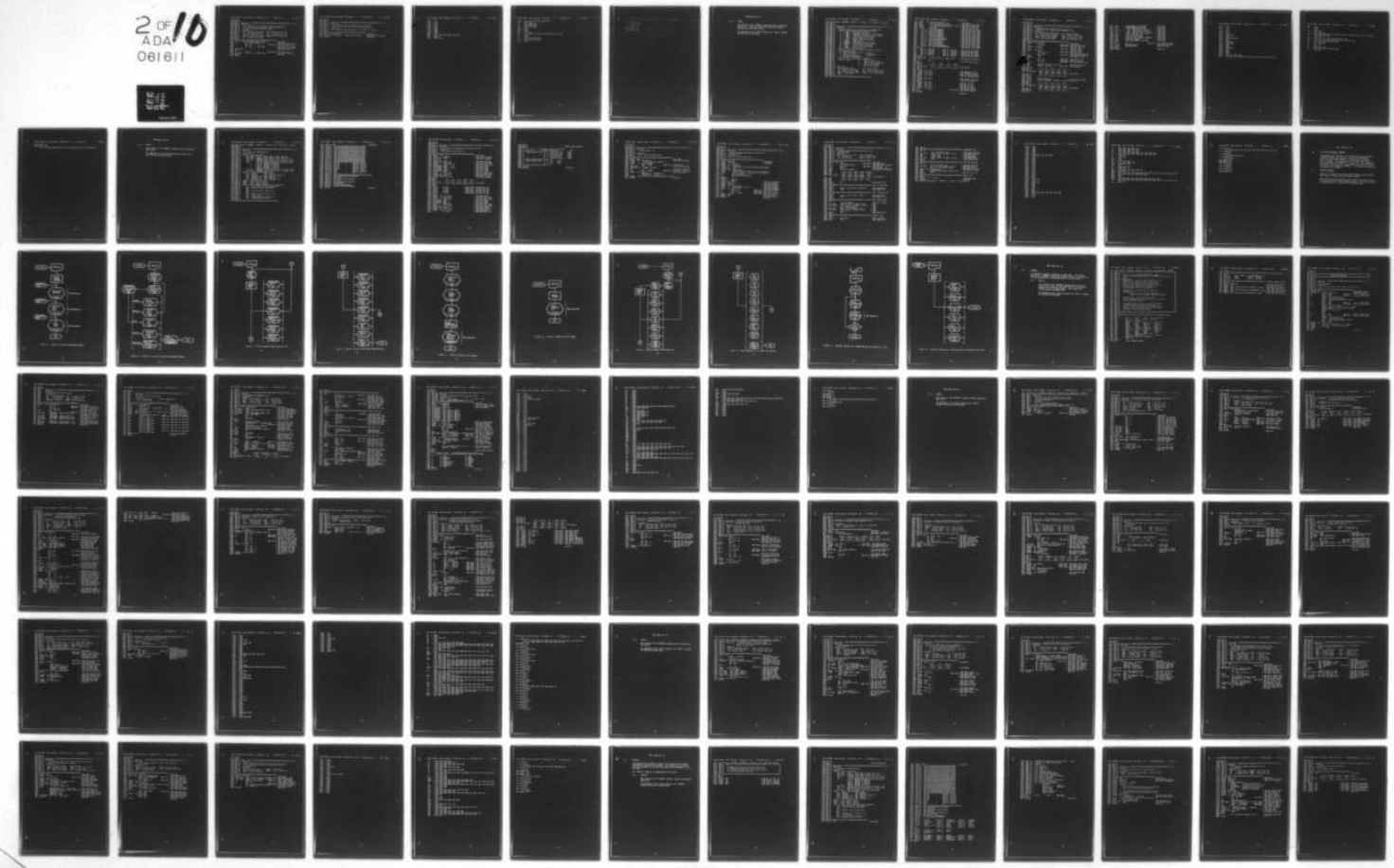
The remainder of this section presents the 'ATH3/I' program listing and reference maps.

```
5000 %## Program > AUTHOR/1  (ATH3/I  )  Version > 2.0  2200 T/VP  780316
5004 REM **
5008 REM **  Subroutine  ********************************************** 1 2 4
5012 REM **  Abstract >
5016 REM **    This routine will select the type of special
5020 REM **    character scanning to be executed.
5024 REM **  Variables >
5028 REM **    W6$ > scaning flag bufr ! RO$() > char input bufr
5032 REM **
5036 REM **********************************************
5040  DEFFN ' 124                                          : REM ENTRY
5044  IF          W6$ = "ZERO"              THEN 5212 : REM SCAN SELECTED
5048  IF          W6$ = "ZEE"               THEN 5212 : REM SCAN SELECTED
5052  IF          W6$ = "BOTH"              THEN 5212 : REM SCAN SELECTED
5056  IF          W6$ = "NULL"              THEN 5212 : REM SCAN SELECTED
5060  REM **
5064  PRINT       HEX(03)                              : REM CLR CRT SCREEN
5068  GOSUB ' 101 (5,1,0,0)                            : REM POSITION CUSOR
5072  SELECT      PRINT 005                            : REM SEL DISPLY MODE
5076  PRINTUSING  5188, " "                            : REM PRINT MENU
5080  PRINTUSING  5192, " "
5084  PRINTUSING  5196, " "
5088  PRINTUSING  5200, " "
5092  SELECT      PRINT 405                            : REM SEL DISPLY MODE
5096  GOSUB ' 101 (2,1,0,0)                            : REM POSITION CUSOR
5100  PRINTUSING  5204, " "
5104  KEYIN       RO$(1), 5108, 5108      : GOTO 5104 : REM RESPONCE WAIT
5108  PRINT       HEX(03)                              : REM CLR CRT SCREEN
5112              W6$ = "NULL"                         : REM SET DEFLT VALUE
5116  IF          RO$(1) <> "1"             THEN 5128 : REM SET '0' SCAN
5120              W6$ = "ZERO"
5124  REM **
5128  IF          RO$(1) <> "2"             THEN 5140 : REM SET 'Z' SCAN
5132              W6$ = "ZEE"
5136  REM **
5140  IF          RO$(1) <> "3"             THEN 5152 : REM SET '0-Z' SCAN
5144              W6$ = "BOTH"
5148  REM **
5152  IF          W6$ = "NULL"              THEN 5212 : REM BRNCH TO EXIT
5156  IF          W6 <> 3                   THEN 5212 : REM NOT LPT OUTPUT
5160              W4$ = "SET PRINTER"                  : REM SET SCAN FLAG
5164  SELECT #5   015                                  : REM SEL LINE PRINTER
5168  SELECT TAPE 07B                                  : REM SEL TAPE DRIVE
5172  GOSUB ' 118                                      : REM INIT PRINTER
5176  REM **
5180  REM **                                             OUTPUT IMAGE FORMATS
5184  REM **
5188  % # Special Character Scanning >   1 - Scan for zero (0 <- /)
5192  % #                               2 - Scan for zee  (Z <- -)
5196  % #                               3 - Scan for both '0' & 'Z'
5200  % #                               4 - NO Special Scanning
5204  % # Enter Scanning Code from Below Table >
5208  REM **
5212  RETURN                                           : REM EXIT
```

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U).
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

2 OF ADA 10
061611



```
5216 REM **
5220 REM **
5224 REM **  Subroutine  ********************************************** 1 2 5
5228 REM **  Abstract >
5232 REM **    This routine will scan search search return buffer
5236 REM **    and insert special character into text buffer in
5240 REM **    location specified by buffer value.
5244 REM **  Variables >
5248 REM **    R4 > lookup tbl row indx  ! R5 > lookup tbl col indx
5252 REM **    X6 > rtrn row indx on 0   ! X7 > rtrn col indx on 0
5256 REM **    X8 > max table row indx   ! X9 > max table col indx
5260 REM **    R2$() > search bufr value ! RO$() > spcl insert char
5264 REM **    R6 > decimal bufr value   !
5268 REM **
5272 REM ************************************************
5276  DEFFN ' 125 (R4,R5,X8,X9,R2$(1),RO$(1))          : REM ENTRY
5280              R6 = FNR(VAL(R2$(1)))                : REM CMPUT DECMAL POS
5284  IF          R6 <> 0                    THEN 5304 : REM NOT END OF TABLE
5288              X6 = R4    :   X7 = R5               : REM SAVE END OF TABLE
5292              R4 = X8    :   R5 = X9               : REM SEND END OF LOOP
5296  GOTO 5312                                        : REM BRNCH TO EXIT
5300  REM **
5304  IF          W6 = 2                     THEN 5312 : REM TTY OUTPUT
5308  MAT COPY    RO$()<1,1> TO W2$()<R6,1>            : REM INSRT SPCL CHAR
5312  RETURN                                           : REM EXIT
5316 REM **
```

```
5316 REM **
5320 REM **
5324 REM **  Subroutine  ********************************************** 1 2 6
5328 REM **  Abstract >
5332 REM **    This routine will print the text string buffer
5336 REM **    X9$() as specified by input string parameters.
5340 REM **  Variables >
5344 REM **    X8 > beging string position ! X9 > length of string
5348 REM **
5352 REM **********************************************
5356  DEFFN ' 126 (X8,X9)                                  : REM ENTRY
5360  $CIO PRINT  #5 (A000    ,R8$(9))X9$()<X8,X9>         : REM PRINT TEXT STRING
5364  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 1 - ATH3/I - # MAP

5104 - 5104
5108 - 5104
5128 - 5116
5140 - 5128
5152 - 5140
5188 - 5076
5192 - 5080
5196 - 5084
5200 - 5088
5204 - 5100
5212 - 5044 5048 5052 5056 5152 5156
5304 - 5284
5312 - 5296 5304

AUTHOR System: Phase 1 - ATH3/I - V MAP

```
R0$(   - 5104 5116 5128 5140 5276 5308
R2$(   - 5276 5280
R4     - 5276 5288 5292
R5     - 5276 5288 5292
R6     - 5280 5284 5308
R8$(   - 5360
W2$(   - 5308
W4$    - 5160
W6     - 5156 5304
W6$    - 5044 5048 5052 5056 5112 5120 5132 5144 5152
X6     - 5288
X7     - 5288
X8     - 5276 5292 5356 5360
X9     - 5276 5292 5356 5360
X9$(   - 5360
```

AUTHOR System: Phase 1 - ATH3/I - ' MAP

???? DEFFN' 101
- 5068 5096
???? DEFFN' 119
- 5172
5040 DEFFN' 124
5270 DEFFN' 125
535c DEFFN' 126

3.2.4 ATH4/I

This section of the 'AUTHOR/1' program contains subroutines 19Ø and 191. These two subroutines represent a general purpose nine-track tape utility.

The remainder of this section contains the 'ATH4/I' program listing and reference maps.

```
5500 %## Program > AUTHOR/1  (ATH4/I  )  Version > 2.0 2200 T/VP  780317
5503 REM **
5506 REM **
5509 REM **  Subroutine  ********************************************** 1 9 0
5512 REM **  Abstract >
5515 REM **    This is a general nine track utility package. This
5518 REM **    Routine will perform the following functions >
5521 REM **      1 - <  BSF > - Backspace one File;
5524 REM **      2 - <  BSR > - Backspace one Record (read ready);
5527 REM **      3 - <  BSW > - Backspace one Record (write ready);
5530 REM **      4 - < CLEAN> - Clean local area on tape;
5533 REM **      5 - <  FSF > - Space Forward one File;
5536 REM **      6 - <  FSR > - Space Forward one Record;
5539 REM **      7 - < READ > - Read Record;
5542 REM **      8 - <REREAD> - Backspace and Read Record;
5545 REM **      9 - <REWIND> - Rewind tape to Load point;
5548 REM **     10 - < WEOF > - Write End Of File mark;
5551 REM **     11 - < WGAP > - Write Inter Record Gap;
5554 REM **     12 - < WRITE> - Write Record.
5557 REM **    A scan for I/O errors will be performed according to
5560 REM **    the following index >
5563 REM **      1 - Data Input/Output errors  (7,12,8)
5566 REM **      2 - Position errors           (1,2,5,6,9,4)
5569 REM **      3 - Control Output errors     (10,11,3)
5572 REM **      4 - No error processing
5575 REM **    A error index is returned in variable >  Z3
5578 REM **    Values of Z3 are >      0  -  Fatal Error
5581 REM **                            1  -  Tape at Load Point
5584 REM **                            2  -  Tape is Write Disabled
5587 REM **                            3  -  End of Tape Sensed
5590 REM **                            4  -  End of File Sensed
5593 REM **  Variables >
5596 REM **    Z1 > Function index          ! Z2 > error index
5599 REM **    Z3 > return error index      ! Z4 > transfer buffer size
5602 REM **    Z5 > suppress error mesg     ! Z6 > device table indx
5605 REM **    Z1$() > micro cmnd buffer    ! Z2$() > error mask buffer
5608 REM **    W2$() > i/o buffer           !
5611 REM **
5614 REM ************************************************
```

```
5617  DEFFN ' 190 (Z1,Z2,Z3,Z4,Z5,Z6)                    : REM ENTRY
5620  DIM         Z1$(13)10,Z2$(12)1,Z3$40               : REM DEFINE WORK BUFRS
5623  DEFFN       Z(Z)=256*VAL(STR(Z1$(Z),9,1))+FNY(Z):  REM DEFINE FUNCTION
5626  DEFFN       Y(Y)=VAL(STR(Z1$(Y),10,1))             : REM DEFINE FUNCTION
5629  SELECT      TAPE 078                               : REM SEL TAPE DEVICE
5632    Z1$( 1) = HEX(6CF344008607)                      : REM DEFINE MICRO CMND
5635    Z1$( 2) = HEX(6CF144008607)                      : REM DEFINE MICRO CMND
5638    Z1$( 3) = HEX(6CF044008607)                      : REM DEFINE MICRO CMND
5641    Z1$( 4) = HEX(6CF744008607)                      : REM DEFINE MICRO CMND
5644    Z1$( 5) = HEX(6CF444008607)                      : REM DEFINE MICRO CMND
5647    Z1$( 6) = HEX(6CF244008607)                      : REM DEFINE MICRO CMND
5650    Z1$( 7) = HEX(6CFB4400C2228607)                  : REM DEFINE MICRO CMND
5653    Z1$( 8) = HEX(6CF84400C2228607)                  : REM DEFINE MICRO CMND
5656    Z1$( 9) = HEX(6CF544008607)                      : REM DEFINE MICRO CMND
5659    Z1$(10) = HEX(6CF944008607)                      : REM DEFINE MICRO CMND
5662    Z1$(11) = HEX(6CF644008607)                      : REM DEFINE MICRO CMND
5665    Z1$(12) = HEX(6CFA4400A2068607)                  : REM DEFINE MICRO CMND
5668    Z1$(13) = HEX(11223344556600000000)              : REM DEFINE MICRO CMND
5671 REM **
5674    Z2$( 1) = HEX(80)     :     Z2$( 2) = HEX(40)    : REM DEFINE ERR MASKS
5677    Z2$( 3) = HEX(20)     :     Z2$( 4) = HEX(10)    : REM DEFINE ERR MASKS
5680    Z2$( 5) = HEX(08)     :     Z2$( 6) = HEX(04)    : REM DEFINE ERR MASKS
5683    Z2$( 7) = HEX(02)     :     Z2$( 8) = HEX(04)    : REM DEFINE ERR MASKS
5686    Z2$( 9) = HEX(02)     :     Z2$(10) = HEX(01)    : REM DEFINE ERR MASKS
5689 REM **
5692  $GIO TAPE   (Z1$(Z1),Z1$(13))   W2$()<1,Z4>        : REM EXECUTE FUNCTION
5695 REM **
5698              Z3 = 0                                 : REM CLR RETURN FLAG
5701 REM **
5704 REM **       !   I/O  !   POS  !   CTL  !  NULL  !
5707 REM **       !--------!--------!--------!--------!
5710  ON Z2 GOTO     5719  ,  5743  ,  573*  ,  5764    : REM BRANCH
5713 REM **       !--------!--------!--------!--------!
5716 REM **
5719  GOSUB ' 191 (7,2)                                 : REM RCVRBL DAT ERR
5722  GOSUB ' 191 (8,9)                                 : REM TRANSMIT DAT ERR
5725  GOSUB ' 191 (8,10)                                : REM DAT OVERFLOW ERR
5728  GOSUB ' 191 (9,10)                                : REM DAT OVR/UNDR FLOW
5731 REM **
5734  GOSUB ' 191 (7,1)                                 : REM IRRCVRBL DAT ERR
5737  GOSUB ' 191 (7,4)                                 : REM WRITE DISABLED
5740 REM **
5743  GOSUB ' 191 (7,3)                                 : REM BEG OF TAPE
5746  GOSUB ' 191 (7,5)                                 : REM END OF TAPE
5749  GOSUB ' 191 (7,6)                                 : REM END OF FILE
5752  GOSUB ' 191 (7,7)                                 : REM DEV NOT READY
5755  GOSUB ' 191 (8,8)                                 : REM MICRO CMND ERR
5758  IF          Z3 = 3                  THEN 5692     : REM REPROCS RECORD
5761 REM **
5764  RETURN                                            : REM EXIT
```

```
5767 REM **
5770 REM **
5773 REM **  Subroutine  ********************************************** 1 9 1
5776 REM **  Abstract >
5779 REM **    To mask the GIO register buffer with error
5782 REM **    mask buffer and display error messages if any.
5785 REM **  Variables >
5788 REM **    Z5 > supress non-fatal msgs  ! Z7 > buffer register
5791 REM **    Z8 > error index to check    ! Z4 > transfer bufr size
5794 REM **    Z9 > characters processed    ! Z1$() > micro cmnd buffer
5797 REM **    Z2$() > error mask buffer    ! Z3$ > non-fatal err msg
5800 REM **    Z3 > return error index      ! Z0 > multi volume flag
5803 REM **
5806 REM ***********************************************
5809  DEFFN ' 191 (Z7,Z8)                                  : REM ENTRY
5812  IF          Z3 <> 0                    THEN 5956 : REM PRIVIOUS ERR
5815  IF          Z7 <> 9                    THEN 5842 : REM NOT COUNT REG
5818  GOSUB ' 191 (7,6)                                  : REM EOF TEST
5821  IF          Z3 = 4                     THEN 5956 : REM EOF RECORD
5824  GOSUB ' 191 (7,5)                                  : REM EOT TEST
5827  IF          Z3 = 3                     THEN 5956 : REM EOT ERROR
5830              Z9 = FNZ(13)                         : REM CMPUT TRANSFR CNT
5833  IF          Z4 <> Z9                   THEN 5881 : REM OVR/UNDR FLOW ERR
5836  GOTO        5956                                 : REM BRANCH TO EXIT
5839 REM **
5842  IF          Z8 <> 5                    THEN 5854 : REM NOT EOT TEST
5845  IF          Z1 <> 9                    THEN 5854 : REM NOT REWIND
5848  IF          Z0 = 99                    THEN 5869 : REM PREVIOUS EOT ERR
5851 REM **
5854  AND         (Z2$(Z8),STR(Z1$(13),Z7,1))          : REM MASK REGISTER
5857  IF          Z2$(Z8) = HEX(00)          THEN 5956 : REM NO ERROR
5860 REM **
5863 REM **       ! 01 F ! 02 F ! 03 I ! 04 I ! 05 I  !
5866 REM **       !------!------!------!------!------!
5869  ON Z8 GOTO    5881 , 5881 , 5914 , 5917 , 5920  ,
                    5926 , 5881 , 5881 , 5881 , 5881  : REM BRANCH
5872 REM **       !------!------!------!------!------!
5875 REM **       ! 06 I ! 07 F ! 08 F ! 09 F ! 10 F  !
5878 REM **
5881  PRINT       HEX(01070C0C0C);                     : REM POSITION CUSOR
5884  PRINT       "Touch 'CONTINUE' 'RETURN' to Attempt Recovery";
5887 REM **
5890 REM **       ! 01 F ! 02 F ! 03 I ! 04 I ! 05 I  !
5893 REM **       !------!------!------!------!------!
5896  ON Z8 GOTO    5908 , 5911 , 5914 , 5917 , 5923  ,
                    5926 , 5929 , 5932 , 5935 , 5938  : REM BRANCH
5899 REM **       !------!------!------!------!------!
5902 REM **       ! 06 I ! 07 F ! 08 F ! 09 F ! 10 F  !
5905 REM **
```

```
5908  STOP        "IRRECOVERABLE DATA ERROR"                : GOTO 5956
5911  STOP        "RECOVERABLE DATA ERROR"                  : GOTO 5956
5914  Z3 = 1      : Z3$ = "BEGINING OF TAPE"                : GOTO 5944
5917  Z3 = 2      : Z3$ = "TAPE WRITE DISABLED"             : GOTO 5944
5920  Z3 = 3      : Z3$ = "END OF TAPE"  : Z0 = 99          : GOTO 5881
5923  STOP        "MOUNT FIRST/NEXT TAPE VOLUME..."         : GOTO 5944
5926  Z3 = 4      : Z3$ = "END OF FILE"                     : GOTO 5944
5929  STOP        "TAPE UNIT NOT READY"                     : GOTO 5956
5932  STOP        "INVALID MICRO COMMAND SEQUENCE"          : GOTO 5956
5935  STOP        "DATA READ ERROR"                         : GOTO 5956
5938  STOP        "DATA OVER/UNDER FLOW ERROR"              : GOTO 5956
5941 REM **
5944  IF          Z5 = 0                        THEN 5956   : REM SUPPRESS MESG
5947  PRINT       HEX(01070C0C0C);                          : REM POSITION CUSOR
5950  PRINT       STR(Z3$,1,40);                            : REM PRINT MSG
5953 REM **
5956  RETURN                                                : REM EXIT
```

AUTHOR System: Phase 1 - ATH4/1 - # MAP

```
5692  - 5758
5719  - 5710
5734  - 5710
5743  - 5710
5764  - 5710
5842  - 5815
5854  - 5842 5845
5869  - 5848
5881  - 5833 5869 5920
5908  - 5896
5911  - 5896
5914  - 5869 5896
5917  - 5869 5896
5920  - 5869
5923  - 5896
5926  - 5869 5896
5929  - 5896
5932  - 5896
5935  - 5896
5938  - 5896
5944  - 5914 5917 5923 5926
5956  - 5812 5821 5827 5836 5857 5908 5911 5929 5932 5935 5938 5944
```

```
W2$(   - 5692
Y      - 5626
Z      - 5623
Z0     - 5848 5920
Z1     - 5617 5692 5845
Z1$(   - 5620 5623 5626 5632 5635 5638 5641 5644 5647 5650 5653 5656 5659 5662
         5665 5668 5692 5854
Z2     - 5617 5710
Z2$(   - 5620 5674 5677 5680 5683 5686 5854 5857
Z3     - 5617 5698 5758 5812 5821 5827 5914 5917 5920 5926
Z3$    - 5620 5914 5917 5920 5926 5950
Z4     - 5617 5692 5833
Z5     - 5617 5944
Z6     - 5617
Z7     - 5809 5815 5854
Z8     - 5809 5842 5854 5857 5869 5896
Z9     - 5830 5833
```

AUTHOR System: Phase 1 - ATH4/1 - ' MAP

5617 DEFFN' 190
5809 DEFFN' 191
- 5719 5722 5725 5728 5734 5737 5743 5746 5749 5752 5755 5818 5824

3.2.5 ATH5/I

This section of the 'AUTHOR/1' program contains subroutines 2ØØ to 2Ø6.

The remainder of this section contains the 'ATH5/I' program listing and reference maps.

```
6000 %## Program > AUTHOR/1  (ATH5/I  )  Version > 2.0  2200 T/VP   780317
6004 REM **
6008 REM **
6012 REM **  Subroutine  ********************************************** 2 0 0
6016 REM **  Abstract >
6020 REM **    To initialize program Tables.
6024 REM **  Variables >
6028 REM **   R4$() > conversion tables
6032 REM **      (01)-(04) > <1,64>   lower case to upper case conv.
6036 REM **      (05)-(08) > <65,64>  upper case to lower case conv.
6040 REM **      (09)-(10) > <129,32> conv binary indx to fiche row char.
6044 REM **      (11) >      <161,16> control character conversion (input)
6048 REM **      (12) >      <177,16> buffer mode character conversions -
6052 REM **           >      <177,4>  HEX(00) to HEX(20)
6056 REM **           >      <181,4>  HEX(00) to HEX(00)
6060 REM **           >      <185,4>  HEX(30) to HEX(20)
6064 REM **           >      <189,4>  HEX(30) to HEX(00)
6068 REM **      (13)-(14) > <193,32> convert mask digits to normal codes
6072 REM **      (15)-(16) > <225,32> conv fiche row char. to binary indx
6076 REM **   R5$() > single character search ids
6080 REM **      (01) > blank          (02) > HEX(FF)
6084 REM **      (03) > HEX(80)        (04) > HEX(5F)<underscor>
6088 REM **      (05) > HEX(3A)<coln>  (06) > HEX(3C) < < >
6092 REM **      (07) > HEX(3E)< > >   (08) > HEX(2E) < . >
6096 REM **      (09) > HEX(20)<blnk>  (10) > HEX(23) < # >
6100 REM **   R6$() > cusor control <vert.>
6104 REM **   R7$() > cusor control <horz.>
6108 REM **   K5$() > cusor contorl <erase>
6112 REM **   R8$() > micro command buffer & output sequences
6116 REM **      (01) > cusor blink w/  overprint char.
6120 REM **      (02) > output of buffer with CRLF
6124 REM **      (03) >
6128 REM **      (04) > warning tone w/ 2 sec wait
6132 REM **      (05) >
6136 REM **      (06) > output 16 line buffer to crt
6140 REM **      (07) > single tone
6144 REM **      (08) > set register 5 with CR or LF
6148 REM **      (09) > command register buffer
6152 REM **
6156 REM ************************************************
```

AUTHOR System: Phase 1 - ATHS/I - 2 0 0

```
6160  DEFFN ' 200                                                    : REM ENTRY
6164  R4$(01)=  HEX(41614262436344644565466647674868)
6168  R4$(02)=  HEX(49694A6A4B6B4C6C4D6D4E6E4F6F5070)
6172  R4$(03)=  HEX(5171527253735474557556765777587B)
6176  R4$(04)=  HEX(59795A7A2020202020202020202020202020)
6180  R4$(05)=  HEX(6141624263436444654566466747684B)
6184  R4$(06)=  HEX(69496A4A6B4B6C4C6D4D6E4E6F4F7050)
6188  R4$(07)=  HEX(7151725273537454755576567757785B)
6192  R4$(08)=  HEX(79597A5A2020202020202020202020202020)
6196  R4$(09)=  HEX(2041424344454647484944A4B4C4D4E4F)
6200  R4$(10)=  HEX(2020202020202020202020202020202020)
6204  R4$(11)=  HEX(010802A7030D03E6202020202020202020)
6208  R4$(12)=  HEX(2000202000002020203020200030202020)
6212  R5$(01)=  HEX(20)          :       R5$(02)= HEX(FF)
6216  R5$(03)=  HEX(80)          :       R5$(04)= HEX(5F)
6220  R5$(05)=  HEX(3A)          :       R5$(06)= HEX(3C)
6224  R5$(07)=  HEX(3E)          :       R5$(08)= HEX(2E)
6228  R5$(09)=  HEX(20)          :       R5$(10)= HEX(23)
6232  INIT(0A)  R6$()            :       R6$(01)= HEX(01)
6236  INIT(09)  R7$()            :       R7$(01)= HEX(0D)
6240  INIT(20)  K5$()            :       K5     = 1
6244  R8$(01)=  HEX(40064260400812114250400811361454160811 46)
6248  R8$(02)=  HEX(4006A000402140OD400A2020)
6252  R8$(03)=  HEX(FFFF)
6256  R8$(04)=  HEX(40071221400712214007)
6260  R8$(05)=  HEX(FFFF)
6264  R8$(06)=  HEX(A000400D1427160840OA)
6268  R8$(07)=  HEX(40074007)
6272  R8$(08)=  HEX(050D14671708050A)
6276  R8$(09)=  HEX(05163F20556677000000)
6280  RETURN                                                         : REM EXIT
```

```
6284 REM **
6288 REM **
6292 REM **  Subroutine  ********************************************** 2 0 1
6296 REM **  Abstract >
6300 REM **    To initialize program input and output devices.
6304 REM **  Variables >
6308 REM **    R2$ > return input value   !
6312 REM **
6316 REM ********************************************
6320  DEFFN ' 201                                          : REM ENTRY
6324  PRINT       HEX(03)                                  : REM CLR CRT SCREEN
6328  PRINTUSING  6492, " "                                : REM PRINT HEADING
6332  PRINT
6336  PRINTUSING  6496, " "                                : REM PRINT HEADING
6340  GOSUB ' 101 (11,1,0,0)                               : REM POSITION CUSOR
6344  SELECT      PRINT 005                                : REM SEL OUTPUT MODE
6348  PRINTUSING  650C, " ","#"                            : REM PRINT MENU
6352  PRINTUSING  6504, " ","#"                            : REM PRINT MENU
6356  PRINTUSING  6508, " ","#"                            : REM PRINT MENU
6360  PRINTUSING  6512, " "                                : REM PRINT MENU
6364  PRINTUSING  6516, " "                                : REM PRINT MENU
6368  SELECT      PRINT 405                                : REM SEL OUTPUT MODE
6372  GOSUB ' 101 (9,1,0,0)                                : REM POSITION CUSOR
6376  PRINTUSING  6520, " ";                               : REM PRINT PROMPT
6380  GOSUB ' 104 (1,"N","5")                              : REM ENTER RESPONCE
6384 REM **
6388 REM **       !  310 !  B10 !  350 !  320 !  B20  !
6392 REM **       !------!------!------!------!-------!
6396  ON R2 GOTO    6408 , 6412 , 6416 , 6420 , 6424    : REM BRANCH
6400 REM **       !------!------!------!------!-------!
6404 REM **
6408  SELECT      #1 310                  : GOTO 6432 : REM SEL DEV 310
6412  SELECT      #1 B10                  : GOTO 6432 : REM SEL DEV B10
6416  SELECT      #1 350                  : GOTO 6432 : REM SEL DEV 350
6420  SELECT      #1 320                  : GOTO 6432 : REM SEL DEV 320
6424  SELECT      #1 B20                  : GOTO 6432 : REM SEL DEV B20
6428 REM **
6432  GOSUB ' 101 (11,1,5,0)                           : REM ERASE MENU
6436  GOSUB ' 101 (13,1,0,0)                           : REM POSITION CUSOR
6440  SELECT      PRINT 005                            : REM SEL OUTPUT MODE
6444  PRINTUSING  6524, " "                            : REM PRINT MENU
6448  PRINTUSING  6528, " "                            : REM PRINT MENU
6452  PRINTUSING  6532, " "                            : REM PRINT MENU
6456  SELECT      PRINT 405                            : REM SEL OUTPUT MODE
6460  GOSUB ' 101 (11,1,0,0)                           : REM POSITION CUSOR
6464  PRINTUSING  6536, " ";                           : REM PRINT PROMT
6468  GOSUB ' 104 (1,"N","1")                          : REM ENTER RESPONCE
6472  GOSUB ' 101 (13,1,3,0)                           : REM ERASE MENU
6476              01 = R2                              : REM STOR DEV TYPE
```

```
6480 REM **
6484 REM **                                                  OUTPUT IMAGE FORMATS
6488 REM **
6492  %  A U T H O R    An Automatic Authoring System          #
6496  %  P H A S E  I      Format Construction                  #
6500  %  STORAGE DEVICE MENU  #  1 - Flopy Drive #1      <310>
6504  %                       #  2 - Flopy Drive #2      <810>
6508  %                       #  3 - Flopy Drive #3      <350>
6512  %                       #  4 - Fixed Plater        <320>
6516  %                       #  5 - Removable Plater    <820>
6520  %  Enter Storage Device Code > #
6524  %  OUTPUT DEVICE MENU   #  1 - CRT      <005>
6528  %                       #  2 - TTY      <011>
6532  %                       #  3 - LPT      <015>
6536  %  Enter Output Device Code > #
6540 REM **
6544  RETURN                                                 : REM EXIT
```

```
6548 REM **
6552 REM **
6556 REM **  Subroutine  ********************************************** 2 0 2
6560 REM **  Abstract >
6564 REM **    To Load Format file.
6568 REM **
6572 REM **  Variables >
6576 REM **    R2$ > returned input value !
6580 REM **
6584 REM ***********************************************
6588  DEFFN ' 202                                          : REM ENTRY
6592  GOSUB ' 101  (15,1,2,40)                             : REM POSITION CUSOR
6596  PRINT        "Format File Index Number > ";
6600  GOSUB ' 104  (4,"A","SAME")
6604  IF           R2$ <> "STOP"            THEN 6620 : REM NOT END
6608  COM CLEAR     R0                                     : REM CLEAR VARIABLES
6612  LOAD DC T #1, "AUTHOR" 10,9999
6616  END                                                  : REM END OF PROGRAM
6620  IF           R2$ <> "SAME"            THEN 6628 : REM DIFRNT FILE NAM
6624  GOSUB ' 205  (R1)                                    : REM LOAD BUFFER
6628               STR(R2$,5,4) = STR(R2$,1,4)
6632               STR(R2$,1,4) = "ATH1"
6636  GOSUB '  102 ("Phase I Data Subprogram",R2$,"9000")
6640  RETURN                                               : REM EXIT
```

```
6644 REM **
6648 REM **
6652 REM **  Subroutine  ********************************************** 2 0 5
6656 REM **  Abstract >
6660 REM **    To Read the Data Subprogram and load into
6664 REM **    Storage buffer.
6668 REM **  Variables >
6672 REM **    R2$ > returned input value
6676 REM **
6680 REM ********************************************
6684  DEFFN ' 205 (R1)                                    : REM ENTRY
6688  DEFFN ' 209 (R1)                                    : REM ENTRY
6692  PRINT       HEX(03)
6696  GOSUB ' 101  (4,20,1,0)
6700  PRINT       "Title Line of Program Loaded:";HEX(0D0A0A)
6704  PRINTUSING   9000, "  "
6708  GOSUB ' 101  (15,1,0,0)
6712  PRINT       "Touch 'ESCAPE' to Override wait";HEX(0D0A)
6716  PRINT       "Format will be displayed in 30 seconds..."
6720  FOR         R0 = 1 TO 2600
6724              KEYIN R0$(1),6732,6732
6728  NEXT        R0
6732  RESTORE     1
6736  PRINT       HEX(03)
6740  INIT(20)    Q2$()                                   : REM CLR TEXT BUFER
6744  GOSUB ' 206 (1)                                     : REM LOAD BUFFER
6748  GOSUB ' 108 (01)                                    : REM PRINT BUFFER
6752  GOSUB ' 101 (16,1,1,40)                             : REM POSITION CUSOR
6756  PRINT       "More Formats (Y/N) > ";                : REM PRINT PROMPT
6760  GOSUB ' 104 (1,"A","Y")                             : REM ENTER RESPONCE
6764  IF          R2$ = "Y"              THEN 6736        : REM LOAD NXT FMT
6768  IF          01 <> 3                THEN 6780        : REM NOT LPT DEV
6772  IF          W4$ <> "TAPE"          THEN 6780        : REM TAPE NOT USED
6776  GOSUB ' 108 (4)                                     : REM PRNT OVRLY
6780  GOSUB ' 202                                         : REM LOAD NXT FILE
6784  RETURN                                              : REM EXIT
```

```
6788 REM **
6792 REM **
6796  REM **  Subroutine  ********************************************* 206
6800  REM **  Abstract >
6804  REM **    To Read the Data Subprogram and load into
6808  REM **    Storage buffer.
6812  REM **  Variables >
6816  REM **    CO > initial bufr indx  ! C1 > row number
6820  REM **    C2 > col number         ! C3 > length of text
6824  REM **    C4 > control index      ! S4$() > input bufr
6828  REM **
6832  REM ***********************************************
6836  DEFFN ' 206  (CO)                                    : REM ENTRY
6840  IF           CO <> 3                        THEN 6856 : REM LOAD BUFR
6844  READ         C1,C2,C3,C4,S4$(1)                     : REM READ DATA PROGRAM
6848  IF           C4 <> 9                        THEN 6844 : REM NOT EOF
6852  GOTO         6872                                   : REM BRACNH TO EOF
6856  READ         C1,C2,C3,C4,S4$(1)                     : REM READ DATA PROGRAM
6860  REM
6864  REM          ! LOWR ! CAPS ! BOLD ! LINE ! UNDRSCR !
6868  REM          !------!------!------!------!---------!
6872  ON C4 GOTO     6904 , 6924 , 6944 , 6988 ,  7008   ,
                     7044 , 7044 , 7044 , 7080 ,  7044   : REM BRANCH
6876  REM          !------!------!------!------!---------!
6880  REM          ! NULL ! NULL ! NULL ! EOF  !  NULL   !
6884  REM
6888  GOTO        7044                                    : REM INSERT TEXT
6892  REM **
6896  REM ********************************************** CONV TO LOWER CASE
6900  REM **
6904  $TRAN        (S4$() <1,C3>, R4$() <65,64>) R        : REM CONVERSION
6908  GOTO         7044                                   : REM INSERT TEXT
6912  REM **
6916  REM ********************************************** CONV TO UPPER CASE
6920  REM **
6924  $TRAN        (S4$() <1,C3>, R4$() <1,64>)  R        : REM CONVERSION
6928  GOTO         7044                                   : REM INSERT TEXT
6932  REM **
6936  REM ********************************************** CONV TO BOLD FACE
6940  REM **
6944               CO = LEN(S4$(1))                       : REM CMPUT LENGTH
6948  MAT COPY     -S4$() <1,CO> TO -S4$()<1,64>          : REM
6952  MAT SEARCH   S4$() <1,64>, <> R5$(1) TO R2$()       : REM
6956               C5 = FNR(VAL(R2$(1)))                  : REM
6960  MAT COPY     R5$() <2,1> TO S4$() <1,1>             : REM
6964  MAT COPY     S4$() <C5,CO> TO S4$() <2,63>          : REM
6968               C3 = C3 + 1                            : REM
6972  GOTO         7044                                   : REM INSERT TEXT
6976  REM **
6980  REM ********************************************** CREATE A LINE
6984  REM **
6988  INIT         (SF) S4$()                             : REM FILL BUFR
6992  GOTO         7044                                   : REM INSERT TEXT
```

```
6996  REM **
7000  REM ********************************************** UNDERSCORE TEXT
7004  REM **
7008                C4 = (C1-1)*64 + C2                  : REM CMPUTE BUF POS
7012  MAT COPY      Q2$() <C4,C3> TO S4$() <1,C3>        : REM LOAD FRM BUFR
7016  AND           (S4$(1),7F)                          : REM MASK TEXT
7020  ADD           (S4$(1),80)                          : REM UNDERSCORE
7024  MAT COPY      S4$() <1,C3>  TO Q2$() <C4,C3>       : REM RELOAD BUFR
7028  GOTO          7044                                 : REM GET NXT LINE
7032  REM **
7036  REM ********************************************** LOAD INTO BUFFER
7040  REM **
7044                C4 = (C1-1)*64 + C2                  : REM CMPUT BUF POS
7048  MAT COPY      S4$() <1,C3> TO Q2$() <C4,C3>        : REM LOAD BUFFER
7052                K1 = C1 - INT((C1-1)/16)*16          : REM CMPUT CRT POS
7056  GOSUB ' 101 (K1,C2,0,0)                            : REM POSITION CUSOR
7060  $GIO          PRINT/005(A000,R8$(9))S4$()<1,C3>
7064  GOTO          6856                                 : REM GET NXT LINE
7068  REM **
7072  REM ********************************************** END OF FILE
7076  REM **
7080  GOSUB ' 101   (1,1,0,0)                            : REM HOME CUSOR
7084  RETURN                                             : REM EXIT
9000  %## Data Subprogram >  ATH101.0   Version > 2.0 09-Sep-77
```

AUTHOR System: Phase 1 - ATHS/1 - # MAP

```
(0010) - 6612
 6408  - 6396
 6412  - 6396
 6416  - 6396
 6420  - 6396
 6424  - 6396
 6432  - 6408 6412 6416 6420 6424
 6492  - 6328
 6496  - 6336
 6500  - 6348
 6504  - 6352
 6508  - 6356
 6512  - 6360
 6516  - 6364
 6520  - 6376
 6524  - 6444
 6528  - 6448
 6532  - 6452
 6536  - 6464
 6620  - 6604
 6628  - 6620
 6732  - 6724
 6736  - 6764
 6780  - 6768 6772
 6844  - 6848
 6856  - 6840 7064
 6872  - 6852
 6904  - 6872
 6924  - 6872
 6944  - 6872
 6988  - 6872
 7008  - 6872
 7044  - 6872 6888 6908 6928 6972 6992 7028
 7080  - 6872
 9000  - 6704
(9999) - 6612
```

```
AUTHOR System: Phase 1 - ATH5/1     - V MAP

C0     - 6836 6840 6944 6948 6964
C1     - 6844 6856 7008 7044 7052
C2     - 6844 6856 7008 7044 7056
C3     - 6844 6856 6904 6924 6968 7012 7024 7048 7060
C4     - 6844 6848 6856 6872 7008 7012 7024 7044 7048
C5     - 6956 6964
K1     - 7052 7056
K5     - 6240
K5$(   - 6240
O1     - 6476 6748 6768
Q2$(   - 6740 7012 7024 7048
R0     - 6608 6720 6728
R0$(   - 6724
R1     - 6624 6684 6688
R2     - 6396 6476
R2$    - 6604 6620 6628 6632 6636 6764
R2$(   - 6952 6956
R4$(   - 6164 6168 6172 6176 6180 6184 6188 6192 6196 6200 6204 6208 6904 6924
R5$(   - 6212 6216 6220 6224 6228 6952 6960
R6$(   - 6232
R7$(   - 6236
R8$(   - 6244 6248 6252 6256 6260 6264 6268 6272 6276 7060
S4$(   - 6844 6856 6904 6924 6944 6948 6952 6960 6964 6988 7012 7016 7020 7024
         7048 7060
W4$    - 6772
```

```
???? DEFFN' 101
       - 6340 6372 6432 6436 6460 6472 6592 6696 6708 6752 7056 7080
???? DEFFN' 102
       - 6636
???? DEFFN' 104
       - 6380 6468 6600 6760
???? DEFFN' 108
       - 6748 6776
6160 DEFFN' 200
6320 DEFFN' 201
6588 DEFFN' 202
       - 6780
6684 DEFFN' 205
       - 6624
6836 DEFFN' 206
       - 6744
6688 DEFFN' 209
```

4.0 DATA BASE MAINTENANCE PROGRAMS

The purpose of this program is to create and maintain an AUTHOR System data base. This data base contains the information recorded in the AUTHOR workbook. The structure of the data base is indexed. This program will allow a data entry clerk to enter and update all elements of the data base. This program will also allow for the data base to be printed onto the line printer. The name of this program is 'AUTHOR/2'. For operating instructions see section 4.0 in Volume I.

4.1 PROGRAM FLOWCHART

Figures 4.1 through 4.8 present program flowcharts which show the major steps of the data base maintenance program.

When appropriate the subroutine name or overlay name will be shown in the lower portion of the flowchart element. This indicates that the named routine will perform the function described.

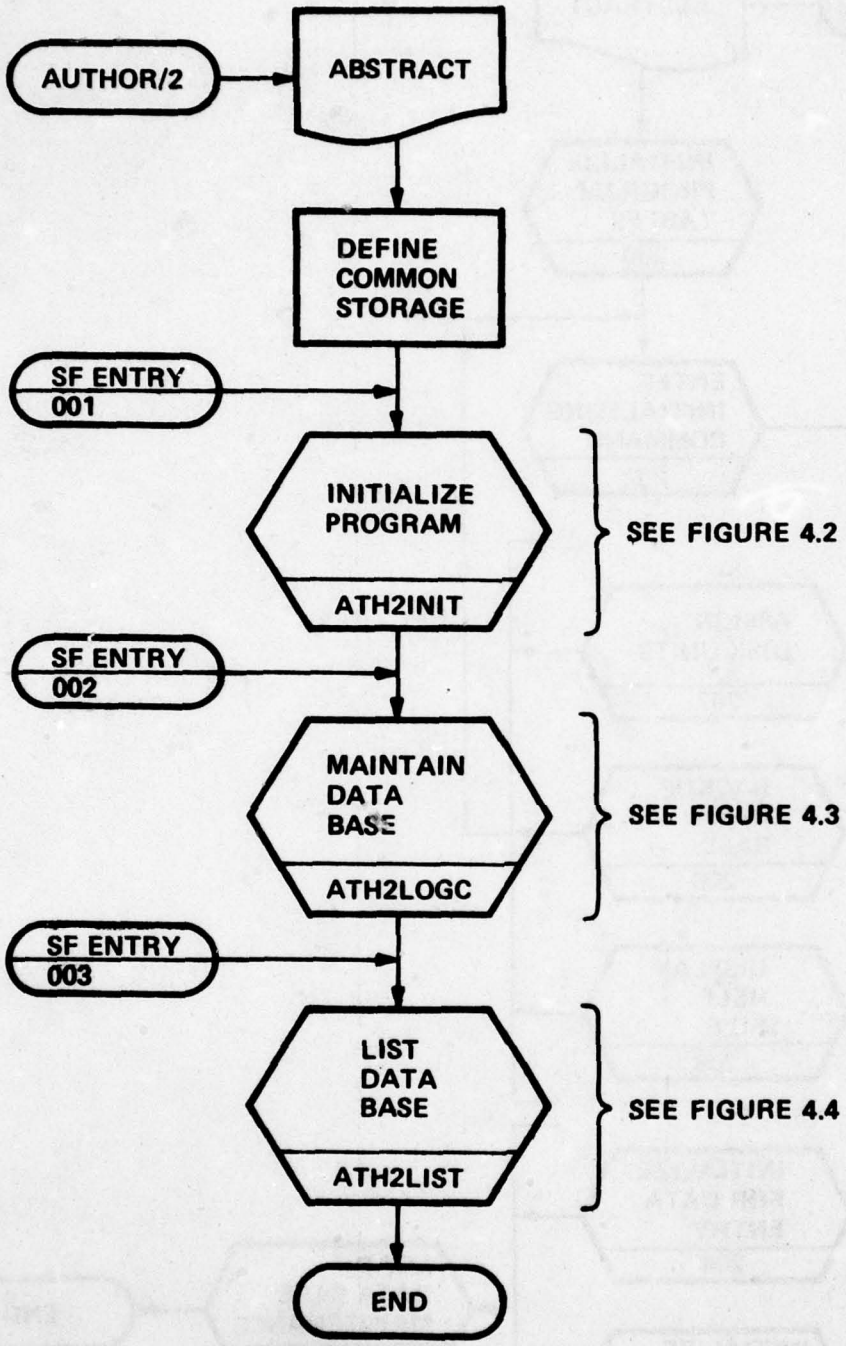


FIGURE 4.1. 'AUTHOR/2' DATA BASE MAINTENANCE PROGRAM

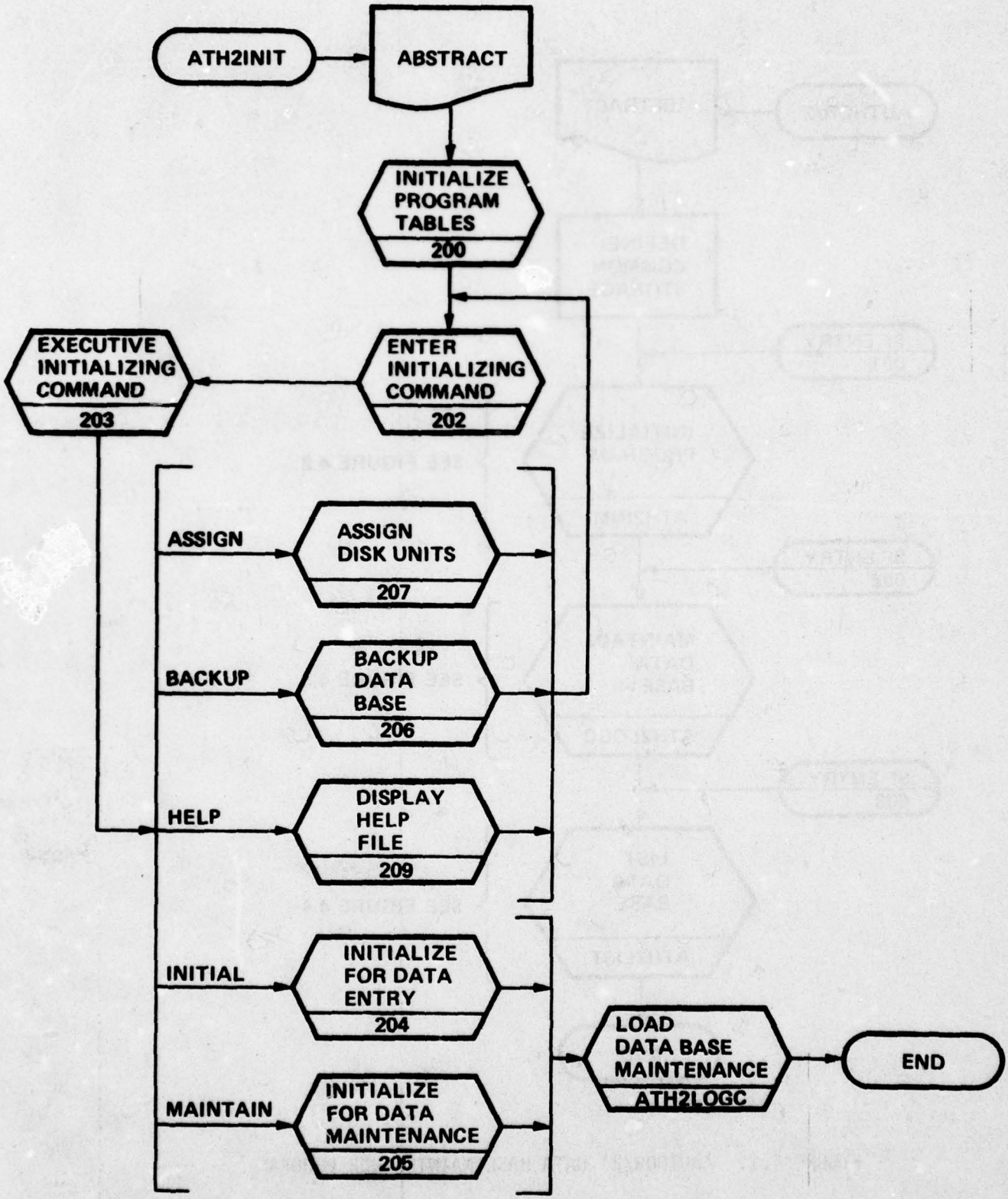


FIGURE 4.2. 'ATH2INIT' INITIALIZE DATA BASE MAINTENANCE PROGRAM

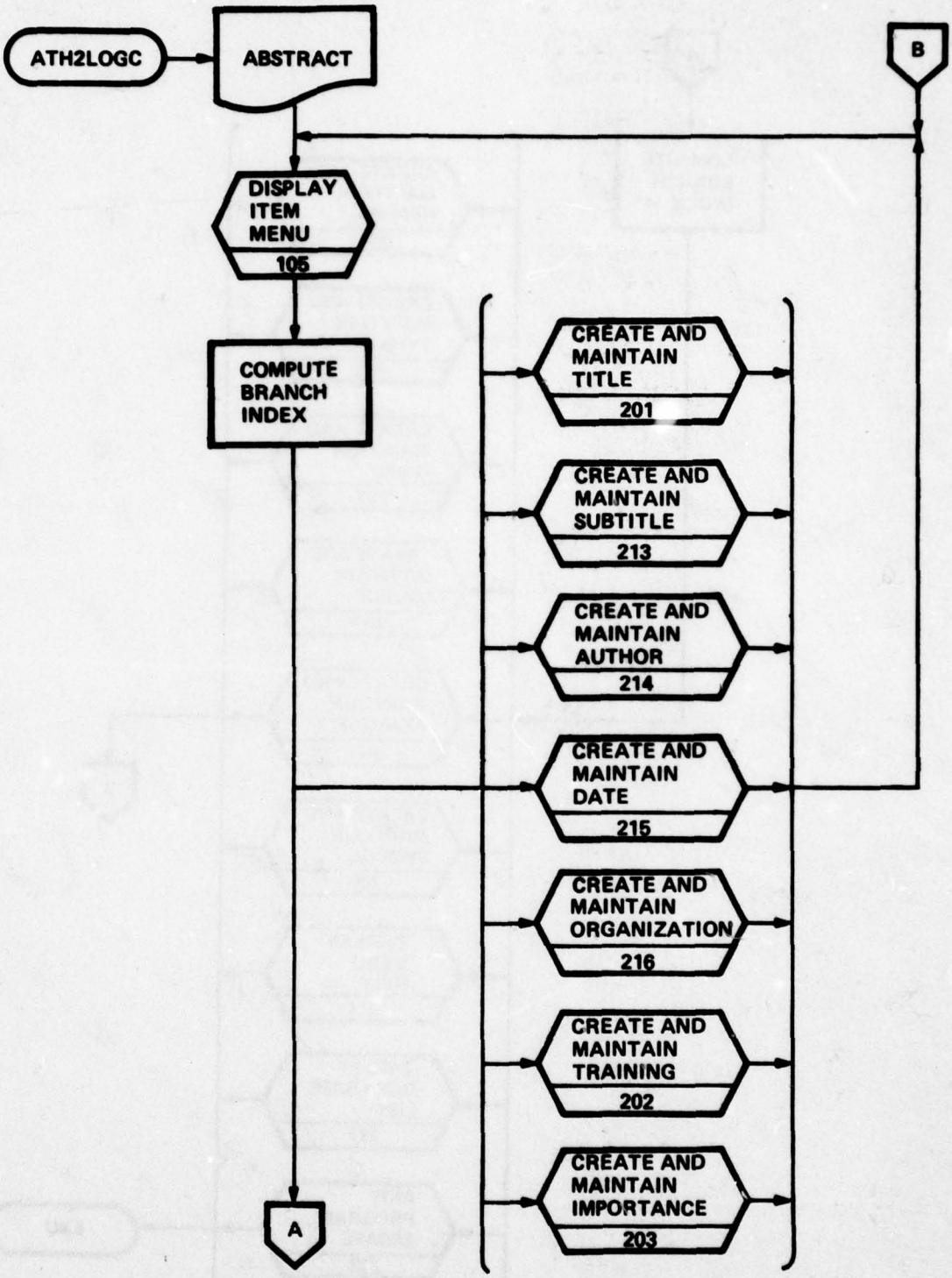


FIGURE 4.3. 'ATH2LOGC' PROGRAM TO MAINTAIN DATA BASE ITEMS

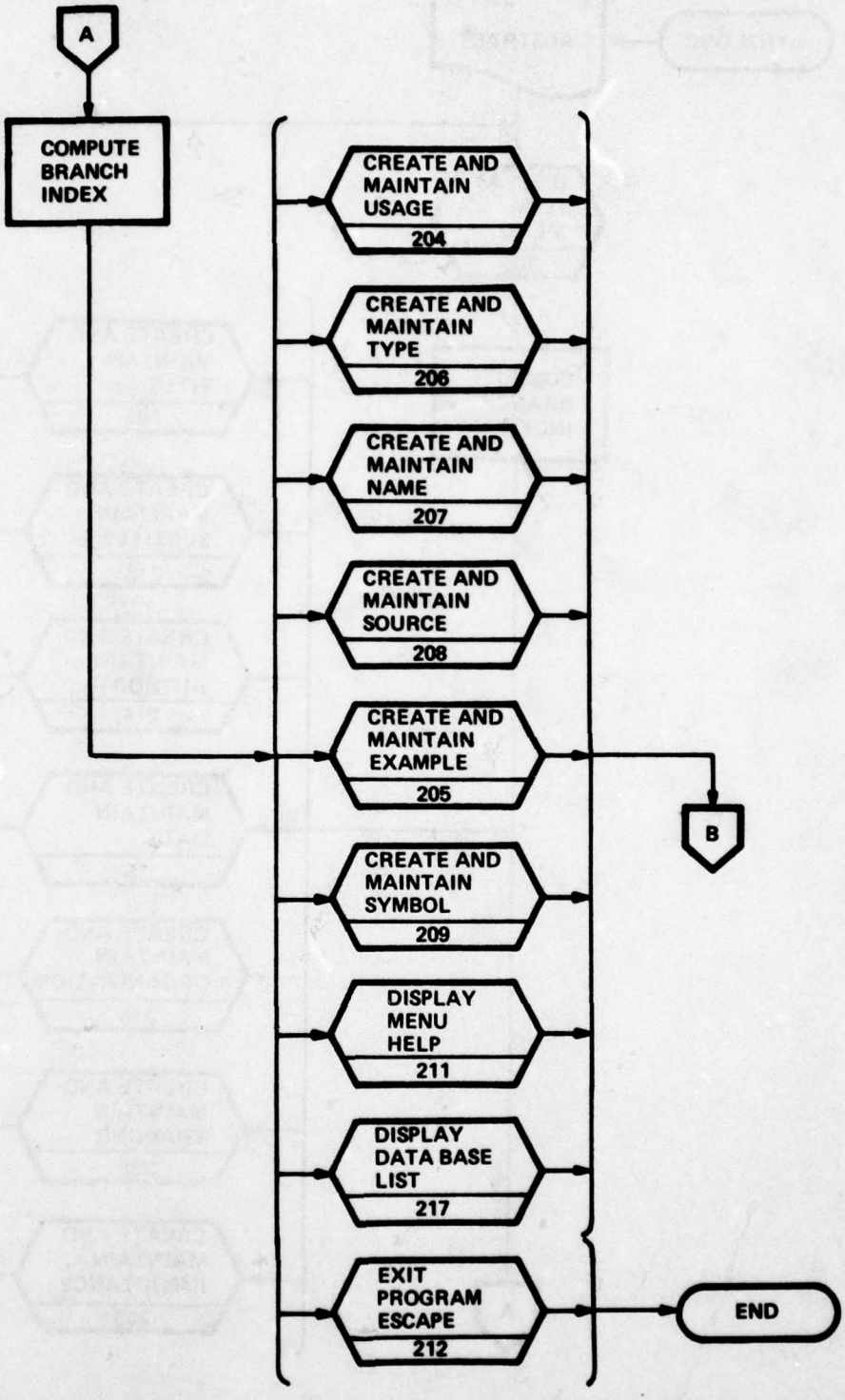


FIGURE 4.3. 'ATH2LOGC' DATA BASE ITEM MAINTENANCE PROGRAM (CONTINUED)

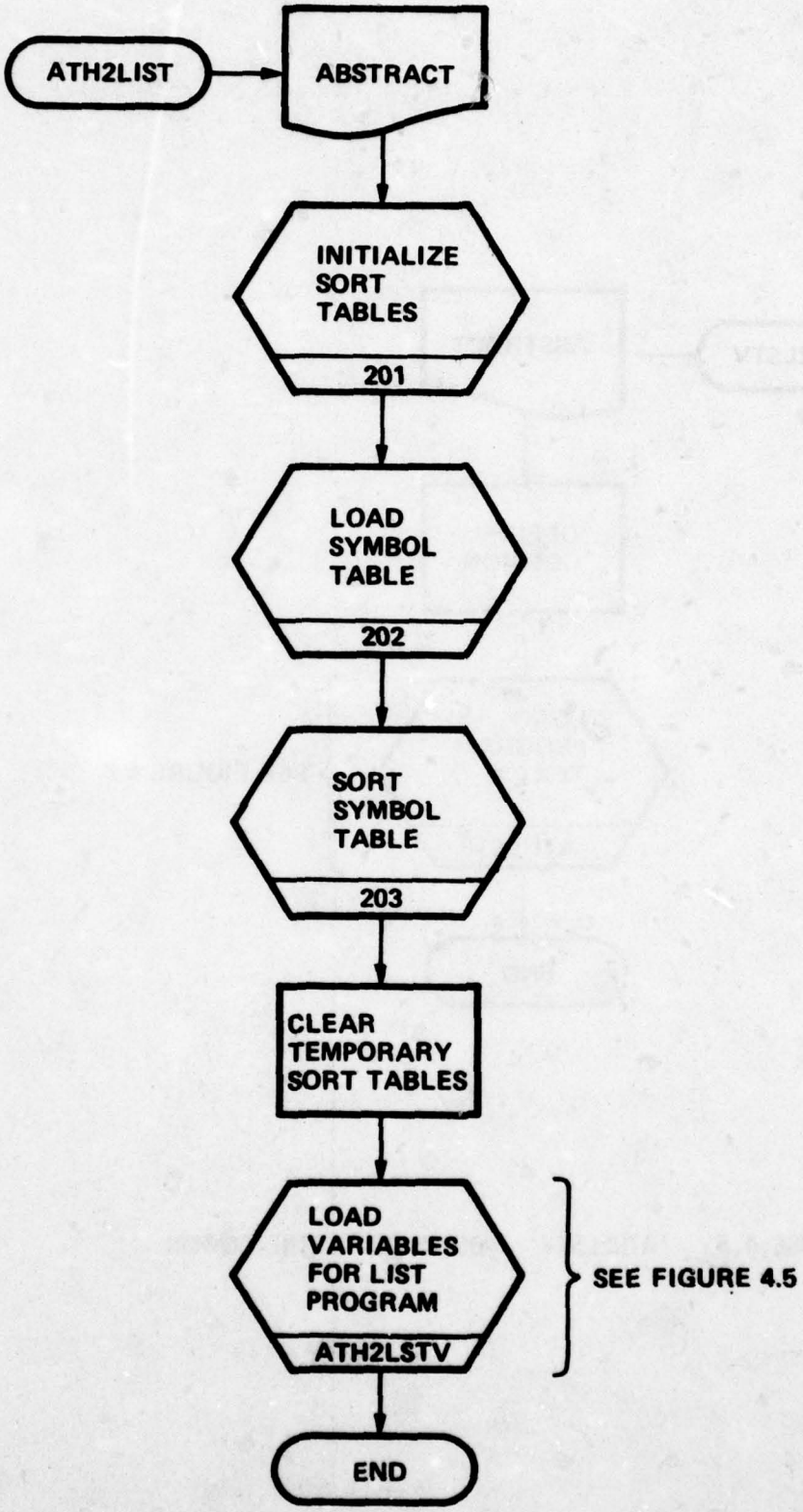


FIGURE 4.4. 'ATH2LIST' PROGRAM TO SORT SYMBOLS

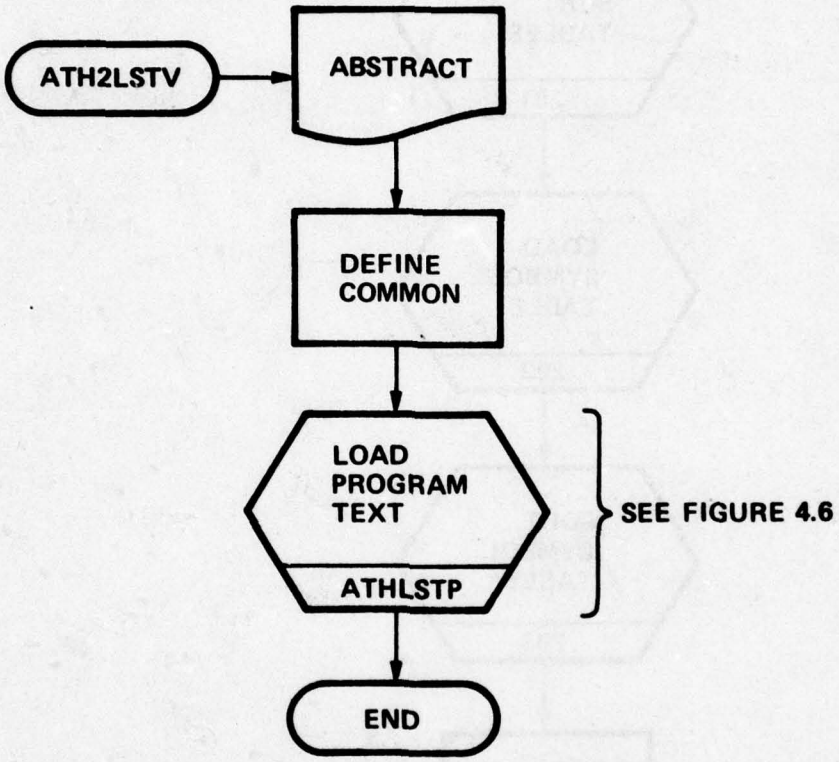


FIGURE 4.5. 'ATH2LSTV' PROGRAM TO DEFINE COMMON

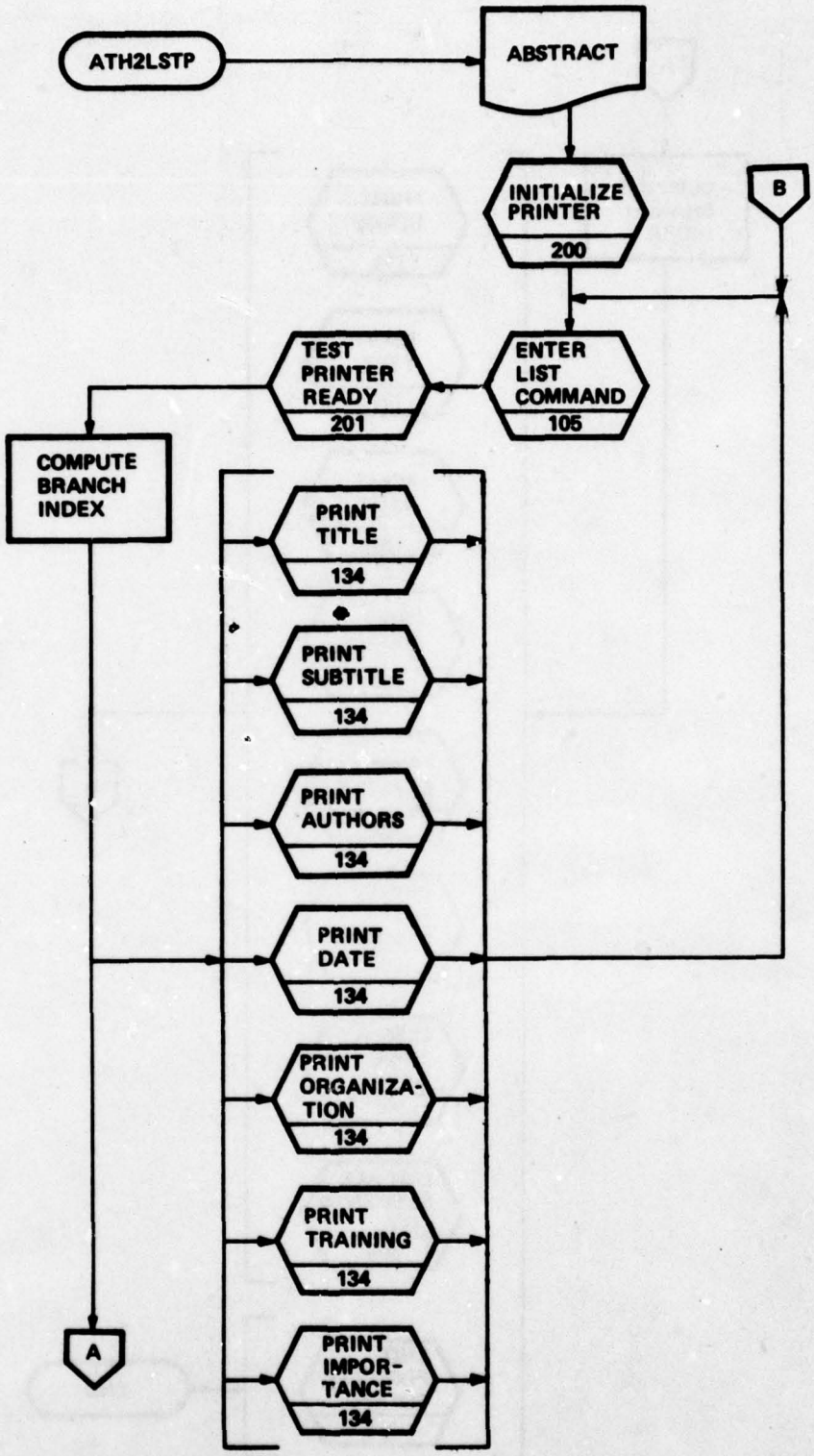


FIGURE 4.6. 'ATH2LSTP' PROGRAM TO LIST DATA BASE ITEMS

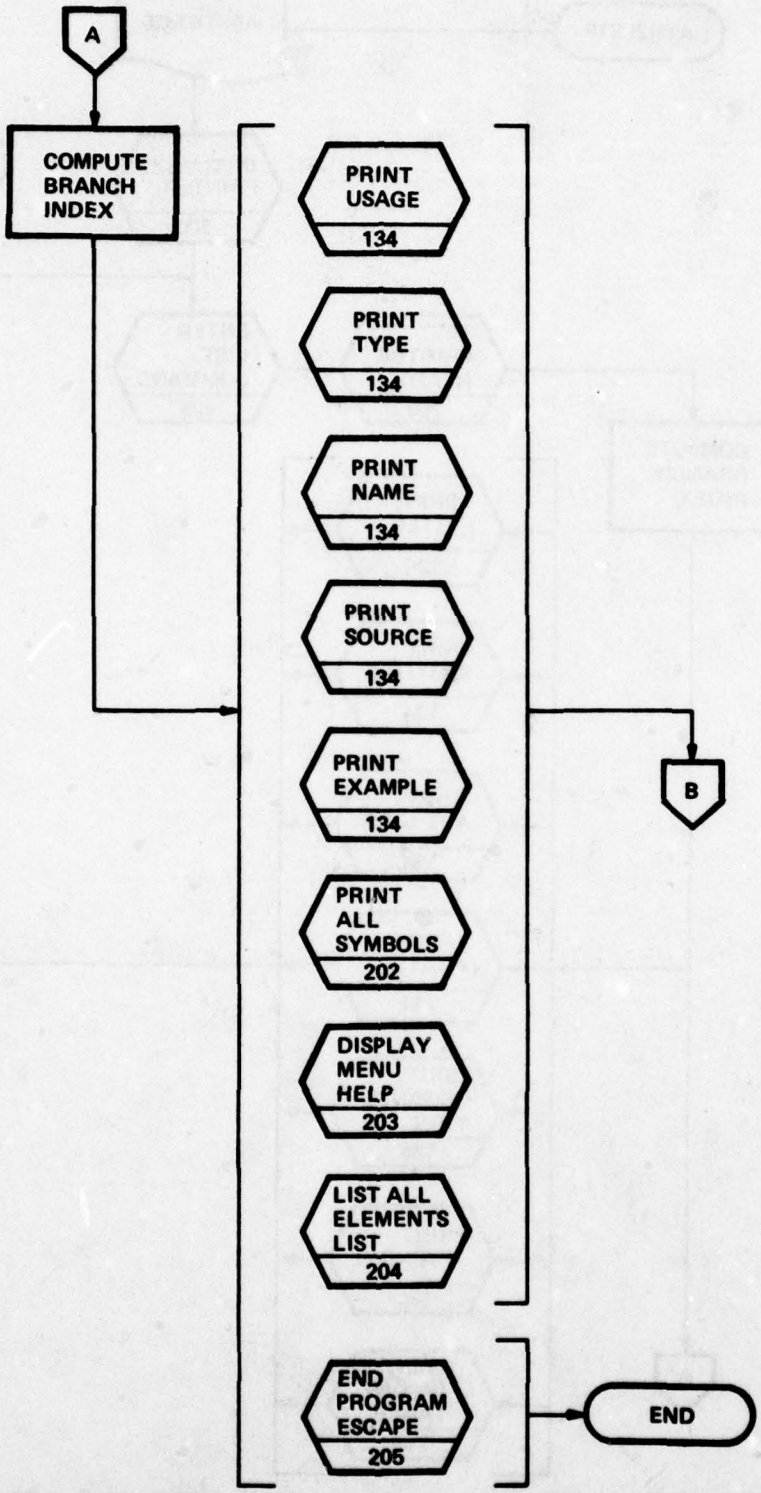


FIGURE 4.6. 'ATH2LSTP' PROGRAM TO LIST DATA BASE ITEMS (CONTINUED)

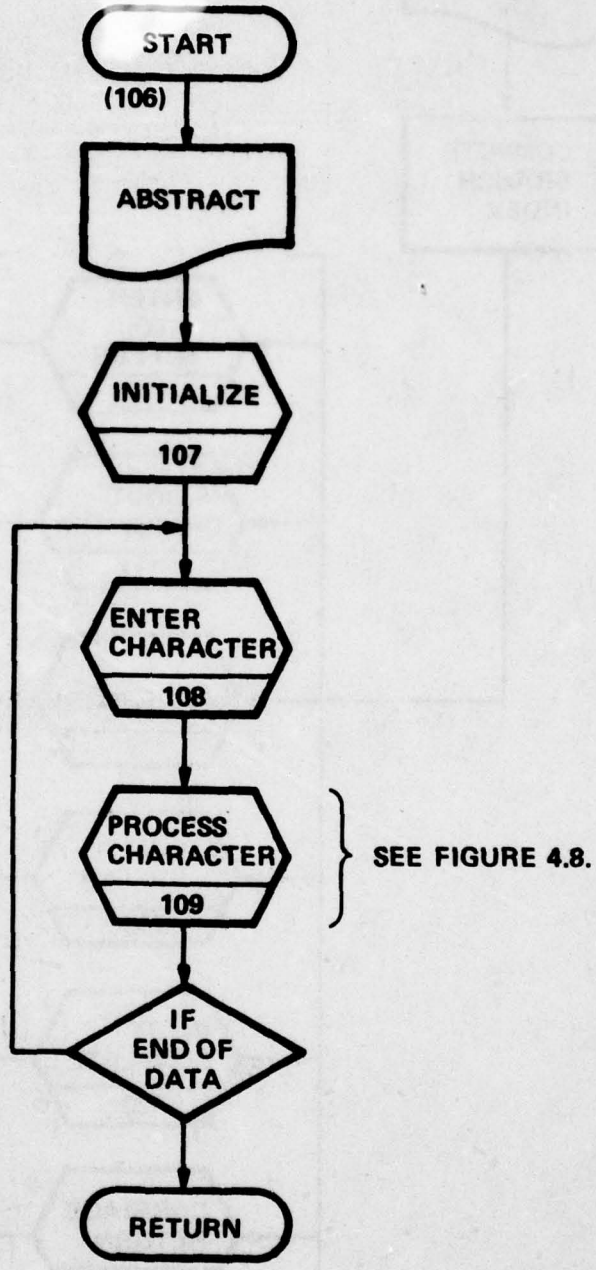


FIGURE 4.7. 'AUTHOR/2' ROUTINE 106--PROGRAM ROUTINE FOR ENTRY/EDIT OF TEXT

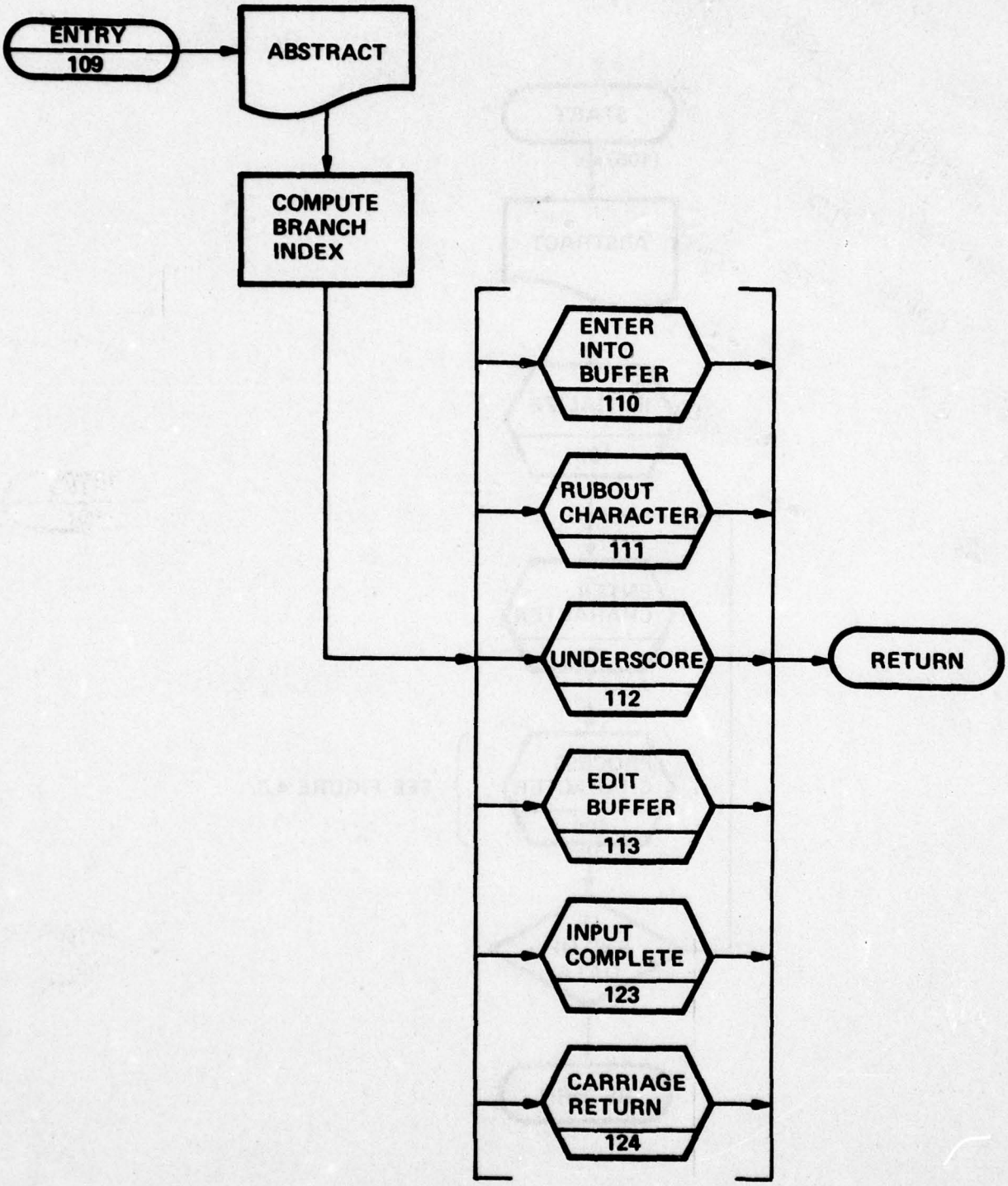


FIGURE 4.8. 'AUTHOR/2' ROUTINE 109--PROGRAM ROUTINE FOR PROCESSING INPUT TEXT

4.2 AUTHOR/2

The 'AUTHOR/2' program is presented in three parts. The routines contained in 'AUTHOR/2' control the storage of data on the disk and also the manipulation of data within the 32 line text buffer.

4.2.1 ATH1/II

This section of the 'AUTHOR/2' program contains the main control logic along with the entry points for loading the maintenance and listing routines. This program also contains subroutines ØØØ to 1Ø5.

The remainder of this section presents the 'ATH1/II' program listing and reference maps.

AUTHOR System: Phase 2 - ATH1/II - START

```
100  %## Program > AUTHOR/2  (ATH1/II )  Version > 2.0 2200 T/VP    780320
104  REM ** Abstract ***************************************************
108  REM **                                                           **
112  REM **    Phase II of the AUTHOR programming system              **
116  REM **    is used to Create and Maintain the AUTHOR Phase II Data **
120  REM **    Base.                                                  **
124  REM ** Program Logic                                            **
128  REM **    AUTHOR/2 will create a Indexed Data Base. The          **
132  REM **    Index will be stored on disk sectors 1-10.             **
136  REM **    Disk sector 0 contains disk creation dates             **
140  REM **    and ID.  The structure of the Data Base will           **
144  REM **    allow access to any item or subitem of                 **
148  REM **    of information stored on the disk.                     **
152  REM **                                                           **
156  REM **    AUTHOR/2 will use two types of Data Entry              **
160  REM **    procedures >                                           **
164  REM **          - Input of a string (1-64 characters)            **
168  REM **          - Input of text ( n pages  32x64 char/page)      **
172  REM **                                                           **
176  REM **    AUTHOR/2 uses a modified form of SOS                   **
180  REM **    (Son Of Stopgap) to allow editing to be performed      **
184  REM **    to the input text buffer.                              **
188  REM **                                                           **
192  REM **    Program Sturcture requires a resident program          **
196  REM **    structure containing initializing and main             **
200  REM **    logic routines also containing program utitlity        **
204  REM **    routines.  Data sturcture routines are loaded          **
208  REM **    according to main program logic.                       **
212  REM **                                                           **
216  REM ***************************************************************
220  REM **
224  REM ********************************************** PROGRAM STORAGE
228  REM **
232  COM          D0, D1   , D2, D3  , D4, D5, D6 , D7, D8, D9
236  COM          E1       , E1$4    , E2$4  , N1$, N2$
240  COM          H8$(23)45, K5$(1)64
244  COM          R0, R1   , R2, R3  , R4, R5, R6 , R7, R8, R9
248  COM          R0$2     , R1$2    , R2$64      , R3$8
252  COM          R4$64    , R5$1    , Q1$(18)2   , Q5$(12)6
256  COM          R6$64    , R7$10   , R8$2       , R9$64
260  COM          R0$(1)1  , R2$(2)2 , R4$(20)16  , R5$(10)1
264  COM          R6$(16)1 , R7$(64)1, R8$(9)24   , R9$(1)64
268  COM          S0$2     , S1$2    , S2$2       , S3$2
272  COM          S4$2     , S5$2    , S6$2       , S7$2
276  COM          Q0(2)    , Q0$(20)1, Q3$(75,6)2
280  COM          Q4$(11)1 , Q6$(10,64)2          , Q7$(5)2
284  COM          Q8$(128)2, Q9$(64)4, R1$(32,63)1
288  DIM          Q2$(1,1)1
292  REM **
300  %##          END OF COMMON STORAGE
```

```
304  REM **
308  REM ********************************************** FUNCTION DEFINITION
312  REM **
316  DEFFN         R(R) = 256*R  + VAL(STR(R2$(1),2,1))
320  INIT(0A)      R6$()         : R6$(1) = HEX(01)
324  INIT(09)      R7$()         : R7$(1) = HEX(0D)
328  REM **
332  REM ********************************************** PROGRAM OVERLAY CONTROL
336  REM **
340  SELECT  PRINT 405                                  : REM SEL OUTPUT DISPLY
344  DEFFN ' 001                                        : REM ENTRY POINT 001
348  GOSUB ' 102 ("Initialization","ATH2INIT","6000")   : REM LOAD INIT ROUTINE
352  DEFFN ' 002                                        : REM ENTRY POINT 002
356  GOSUB ' 102 ("Data Maintenance","ATH2LOGC","6000"): REM LOAD ENTRY ROUTNE
360  DEFFN ' 003                                        : REM ENTRY POINT 003
364  GOSUB ' 102 ("Data Listing","ATH2LIST","1500")     : REM LOAD LIST ROUTINE
```

```
368 REM **
372 REM **
376 REM ************************************************************************
380 REM **                        LABELED SUBROUTINES                           **
384 REM ************************************************************************
388 REM **
392 REM **
396 REM **  Subroutine  ************************************************ 0 0 0
400 REM **  Abstract >
404 REM **    Error Recovery.
408 REM **  Variables >
412 REM **    E1$ > two byte error code ! E2$ > line in error
416 REM **
420 REM ************************************************
424  DEFFN ' 000                                                : REM NORMAL ENTRY
428  ON ERROR      E1$,E2$  GOTO 432                            : REM ERROR ENTRY
432  IF            E1$ <> "80"              THEN 456            : REM T  ERROR CHECK
436  IF            E1$ <> "D82"             THEN 456            : REM VP ERROR CHECK
440                GOSUB ' 101 (16,24,1,40)
444                PRINT " FILE NOT IN CATALOG"
448                $GIO    PAUSE TONE/005(R8$(4),R8$(9))
452                GOTO    512
456  IF            E1$ <> "29"              THEN 488            : REM T  ERROR CHECK
460  IF            E1$ <> "X73"             THEN 488            : REM VP ERROR CHECK
464                GOSUB ' 101 (16,24,1,40)
468                PRINT " Error > ";E1$;" At Line > ";E2$;
472                PRINT HEX(0D0A);" Invalid Data Format"
476                GOSUB ' 101 (1,1,5,0)
480                STOP
484                END
488  IF            E1$ <> "27"              THEN 512            : REM T  ERROR CHECK
492  IF            E1$ <> "X70"             THEN 512            : REM VP ERROR CHECK
496                GOSUB ' 101 (16,24,1,40)
500                PRINT " Insufficient Data "
504                $GIO    PAUSE TONE/005(R8$(4),R8$(9))
508                GOTO    512
512  GOSUB ' 101   (15,1,2,40)
516  PRINT         " Error > ";E1$;" At Line > ";E2$;
520  PRINT         HEX(0D0A);" Correction Required"
524  GOSUB ' 101   (1,1,5,0)
528  STOP
532  END                                                        :  REM EXIT
```

```
536  REM **
540  REM **
544  REM       Subroutine > ********************************************  1 0 1
548  REM       Function   >
552  REM        Cusor Address and Screen erase
556  REM       Varaibles  >
560  REM        K1     > row position !  K3     > lines to erase
564  REM        K2     > col position !  K4     > char(s) erase
568  REM        R6$() > cusor vert    !  R7$() > cusor horz
572  REM        K5,K5$() > line ers !  R8$() > micro cmnd bufr
576  REM
580  REM ***********************************************
584  DEFFN '  101  (K1,K2,K3,K4)                              : REM ENTRY
588  IF            K3 = 0                        THEN 632    : REM ERASE NO LINE
592  IF            K4 <> 0                       THEN 600    : REM CHAR TO ERS SPCFD
596                K4 = 65 - K2                              : REM ERS REST OF LINE
600  FOR           K5 = 1 TO K3                              : REM BEGIN LOOP K5
604  IF            K5 <> 1                       THEN 616    : REM NOT FIRST LINE
608  $GIO ROW      /005(A000  ,R8$(9))R6$()<1,K1>            : REM POSITION CUSOR
612                GOTO 620                                  : REM BRANCH TO POS COL
616  $GIO ROW      /005(A000  ,R8$(9))R6$()<2,1>             : REM POSITION CUSOR LF
620  $GIO COL      /005(A000  ,R8$(9))R7$()<1,K2>            : REM POSITION CUSOR
624  $GIO ERASE    /005(A000  ,R8$(9))K5$()<1,K4>            : REM ERASE TEXT
628  NEXT          K5                                        : REM CONTINUE LOOP K5
632  $GIO ROW      /005(A000  ,R8$(9))R6$() <1,K1>           : REM REPOSITION CUSOR
636  $GIO COL      /005(A000  ,R8$(9))R7$() <1,K2>           : REM REPOSITION CUSOR
640  RETURN                                                  : REM EXIT
```

```
644  REM **
648  REM **
652  REM       Subroutine > ********************************************  1 0 2
656  REM       Function   >
660  REM         To load an over lay Routine.
664  REM       Variables  >
668  REM         L1$ > description        !  L3$ > begining line
672  REM         L2$ > overlay name       !
676  REM
680  REM ********************************************
684  DEFFN '   102   (L1$,L2$,L3$)                          : REM ENTRY
688  PRINT           HEX(03)                                : REM CLEAR CRT
692  PRINT           "A U T H O R   S Y S T E M"            : REM PRINT TITLE
696  PRINT           HEX(0D0A);"      P H A S E   III"      : REM PRINT SUBTITLE
700  GOSUB '   101   (5,10,0,0)                            : REM POSITION CUSOR
704  PRINT     "System Now Loading >>> ";L1$;" Routine":    REM PRINT DESCRIPTION
708  GOSUB '   101   (7,10,0,0)                            : REM POSITION CUSOR
712  PRINT           "Program being Loaded >>> ";L2$        : REM PRINT PROG NAME
716  IF              L3$ <> "0100"                THEN 724  : REM LOAD AT LINE 0100
720  LOAD            DC T #1, L2$ 100
724  IF              L3$ <> "1500"                THEN 732  : REM LOAD AT LINE 1500
728  LOAD            DC T #1, L2$ 1500
732  IF              L3$ <> "4000"                THEN 740  : REM LOAD AT LINE 4000
736  LOAD            DC T #1, L2$ 4000
740  IF              L3$ <> "6000"                THEN 748  : REM LOAD AT LINE 6000
744  LOAD            DC T #1, L2$ 6000
748  IF              L3$ <> "7000"                THEN 756  : REM LOAD AT LINE 7000
752  LOAD            DC T #1, L2$ 7000
756  IF              L3$ <> "9000"                THEN 764  : REM LOAD AT LINE 9000
760  LOAD            DC T #1, L2$ 9000
764  GOSUB ' 000                                           : REM ERR PROCESSING
768  RETURN                                                : REM EXIT
```

```
772  REM **
776  REM **
780  REM **  Subroutine  ********************************************** 1 0 4
784  REM **  Abstract >
788  REM **    Accept ALPHA and NUMERIC Input.
792  REM **  Varaibles  >
796  REM **    RO$() > entry buffer      ! R1  > max length
800  REM **    R1$   > 'A' or 'N' or 'E'! R2  > return <num>
804  REM **    R2$   > return <alph>     ! R3  > char pointer
808  REM **    R6$   > initial value     ! R9  > scratch count
812  REM **
816  REM ***********************************************
820   DEFFN ' 104   (R1,R1$,R6$)                                  : REM ENTRY
824   INIT(3F)      R9$()                                         : REM INIT CHAR BUFR
828   $GIO PRINT    /005 (A000,R8$(9))R9$()<1,R1>                 : REM PRNT CUSOR BKGRND
832   INIT(08)      R9$()                                         : REM INIT CHAR BUFR
836   $GIO PRINT    /005 (A000,R8$(9))R9$()<1,R1>                 : REM PRNT CUSOR BKGRND
840   $GIO          /005 (1136,R8$(9))                            : REM INIT REGISTER
844                 R9  =  1                                      : REM PRNT POS INDX
848                 R3  =  1                                      : REM BUFR POS INDX
852   INIT          (20) R2$                                      : REM CLR BUFR
856  REM **
860   KEYIN         RO$(1), 860, 860                              : REM CLR INPUT BUFR
864                 STR(R8$(9),5,1) = STR(R2$,R9,1)               : REM STOR OUT CHAR
868   IF            R1$ <> "E"                        THEN 876    : REM ECHO CHARACTR
872                 STR(R8$(9),5,1) = HEX(20)                     : REM SET NO ECHO
876   KEYIN         RO$(1), 916, 892                              : REM INPUT CHAR
880   $GIO          INPUT WAIT/005(R8$(1),R8$(9))                 : REM PRNT CUSOR
884   GOTO          876                                           : REM CONT INPT WAIT
888  REM **
892   PRINT         HEX(0808);                                    : REM FNT KEY ENTRY
896   INPUT         R2$                                           : REM REENTER SPL KEY
900                 R3 = 5                                        : REM SET LENGTH
904                 STR(R2$,R3) = HEX(20)                         : REM CLR BUFR END
908   GOTO          1136                                          : REM CONV IN DATA
912  REM **
916   IF            RO$(1) < HEX(08)                THEN 936      : REM INVALID INPT
920   $TRAN         (RO$()<1,1>, R4$()<161,10>) R                 : REM CONV CTL CHARS
924   IF            RO$(1) > HEX(7F)                THEN 936      : REM INVALID INPT
928   IF            RO$(1) > HEX(1F)                THEN 956      : REM VALID CHAR
932   IF            RO$(1) < HEX(04)                THEN 956      : REM VALID CTL CHAR
936   $GIO          TONE/005(R8$(7),R8$(9))                       : REM SOUND BELL
940   GOTO          876                                           : REM RE-ENTER CHAR
944  REM **
948  REM **                    ! RUBOUT ! UNDERSCORE ! ESCAPE !
952  REM **                    !--------!------------!--------!
956   ON VAL(RO$(1)) GOTO        1028   ,    1072    ,  1096   : REM BRANCH
960  REM **                    !--------!------------!--------!
```

```
964  REM **
968  REM ********************************************** Enter Character
972  REM **
976  IF          R3 = R1 + 1                    THEN 936  : REM BUFR FULL
980              STR(R2$,R3,1) = R0$(1)                   : REM MOVE TO BUFR
984              R3 = R3 + 1                              : REM INCR BUFR INDX
988  IF          R3 = 2                         THEN 1000 : REM NOT FRST CHAR
992              R9 = R9 + 1                              : REM INCR DSPLY INDX
996  PRINT       HEX(09);                                 : REM POSITION CUSOR
1000  IF          R1$ <> "E"                     THEN 1008 : REM ECHO CHARACTR
1004             R0$(1) = HEX(20)                         : REM SET NO ECHO
1008  PRINT       R0$(1);HEX(08);                          : REM PRNT CHAR
1012  GOTO        864                                      : REM FECTH NXT CHAR
1016  REM **
1020  REM ********************************************** Rubout Character
1024  REM **
1028  IF          R3 = 1                          THEN 864 : REM BUFR EMPTY
1032  PRINT       HEX(3F08);                               : REM POSITION CUSOR
1036             R3 = R3 - 1                              : REM DCRMNT BUF INDX
1040             STR(R2$,R3,1) = HEX(20)                  : REM CLR FRM BUFR
1044  IF          R9 = 1                         THEN 1056 : REM CUSOR POSITIONED
1048             R9 = R9 - 1                              : REM DCRMNT DSP INDX
1052  PRINT       HEX(08);                                 : REM POSITION CUSOR
1056  GOTO        864                                      : REM FETCH NXT CHAR
1060  REM **
1064  REM ********************************************** Underscore char.
1068  REM **
1072  AND         (STR(R2$,R9,1),7F)                       : REM MASK CHAR
1076  ADD         (STR(R2$,R9,1),80)                       : REM UNDERSCORE
1080  GOTO        864                                      : REM FETCH NXT CHAR
1084  REM **
1088  REM ********************************************** String Entered
1092 REM **
1096  IF          R3 <> 1                        THEN 1124 : REM BUFR NOT EMPTY
1100             R2$ = R6$                                : REM INSRT DFLT VAL
1104  IF          R1$ <> "E"                     THEN 1112 : REM ECHO DFLT VAL
1108             R6$ = " "                                : REM CLR DFLT VAL
1112  PRINT       HEX(08);                                 : REM POSITION CUSOR
1116  GOTO        1132                                     : REM BRNCH TO PRNT
1120  REM **
1124             STR(R2$,R3) = HEX(20)                    : REM CLR REST OF BUF
1128             R6$ = R2$                                : REM MOVE TO PRNT BUF
1132  PRINT       HEX(09);STR(R6$,R3,ABS(R1-R3)+1);        : REM PRNT STRNG
1136  IF          R1$ <> "N"                     THEN 1172 : REM RET ALPHA VAL
1140  IF          NUM(R2$) = 64                  THEN 1168 : REM VALID NUM FMT
1144             R2  =  R1                                : REM SET CUSOR INDX
1148  IF          R3 <> R1+1                     THEN 1156 : REM INDX IS OK
1152             R2 =  R1+2                               : REM RESET INDX
1156  INIT(08) R9$(): $GIO/005(A000,R8$(9))R9$()<1,R2>    : REM PRNT CUSOR BKGRND
1160  $GIO        TONE/005(R8$(7),R8$(9))                  : REM ERR TONE
1164  GOTO        824                                      : REM REENTER STRNG
1168  CONVERT     R2$ TO R2                                : REM SET NUMERIC VAL
1172  RETURN                                               : REM EXIT
```

```
1176 REM **
1180 REM **
1184  REM **  Subroutine  ********************************************** 1 0 5
1188  REM ###  Function   >
1192  REM ###    Display program directives and accept input.
1196  REM ###  Varaiables  >
1200  REM ###    RO$() > entry buffer          ! RO   > value of input
1204  REM ###    Q3$() > command memory bufr !
1208  REM **********************************************
1212  DEFFN ' 105 (RO,R2$)                                  : REM ENTRY
1216  SELECT      PRINT 005                                 : REM SEL CONSOLE MODE
1220  PRINT       HEX(03)                                   : REM CLR CONSOLE SCREN
1224  PRINTUSING  1352, R2$                                 : REM PRINT MENU
1228  PRINTUSING  1356, " "
1232  PRINTUSING  1360, GO$(1), GO$(9)
1236  PRINTUSING  1364, GO$(2), GO$(10)
1240  PRINTUSING  1368, GO$(3), GO$(11)
1244  PRINTUSING  1372, GO$(4), GO$(12)
1248  PRINTUSING  1376, GO$(5), GO$(13)
1252  PRINTUSING  1380, GO$(6), GO$(14)
1256  PRINTUSING  1384, GO$(7), GO$(15)
1260  PRINTUSING  1388, GO$(8), GO$(16)
1264  SELECT      PRINT 405                                 : REM SEL CONSOLE MODE
1268  GOSUB ' 101 (15,1,0,0)                                : REM POSITION CUSOR
1272  PRINT       "Enter Command > "                        : REM PRINT PROMPT
1276  GOSUB ' 104 (16,"A",N1$)                              : REM INPUT RESPONCE
1280              RO = LEN(R2$)                             : REM CMPUT CMND LEN
1284  IF          RO <= 2                       THEN 1292   : REM LEN WITHN LIMITS
1288              RO  = 2                                   : REM SET TO MAX LEN
1292  AND         (STR(R2$,1,RO),DF)                        : REM FORCE TO UPPR CAS
1296  MAT SEARCH  Q1$(), =STR(R2$,1,RO) TO R2$() STEP 2:    REM SRCH KEYWRD BUFR
1300  IF          R2$(2) <> HEX(0000)           THEN 1308   : REM INVALID COMMAND
1304  IF          R2$(1) <> HEX(0000)           THEN 1332   : REM KEY FOUND
1308  GOSUB ' 101 (16,1,0,0)                                : REM KEY NOT FOUND
1312  PRINT       " INVALID COMMAND!!     RE-ENTER"         : REM PRINT ERR MSG
1316  $GIO TONE   /005 (R8$(4),R8$(9))                      : REM PAUSE TONE
1320  GOSUB ' 101 (16,1,1,0)                                : REM ERASE ERR MSG
1324  GOTO        1268                                      : REM REENTER COMMAND
1328 REM **
1332              RO = INT(FNR(VAL(R2$(1)))/2) + 1          : REM CMPUT CMND INDX
1336              GO$(RO) = HEX(2A)                         : REM SET MENU FLAG
1340 REM **
1344 REM **                                                   OUTPUT IMAGE FORMATS
1348 REM **
1352  %    COMMAND SUMMARY - ########################################
1356  % # ------------------------------------------------------------
1360  %          #  TITLE                          #  TYPE
1364  %          #  SUBTITLE                       #  NAME
1368  %          #  AUTHORS                        #  SOURCE
1372  %          #  DATE                           #  EXAMPLE
1376  %          #  ORGANIZATION                   #  SYMBOL
1380  %          #  TRAINING                       #  HELP
1384  %          #  IMPORTANCE                     #  LIST
1388  %          #  USAGE                          #  ESCAPE
1392 REM **
1396  RETURN                                                : REM EXIT
```

```
AUTHOR System: Phase 2 - ATH1/I1   - # MAP

 0100  - 0720
 0432  - 0428
 0456  - 0432 0436
 0488  - 0456 0460
 0512  - 0452 0488 0492 0508
 0600  - 0592
 0616  - 0604
 0620  - 0612
 0632  - 0588
 0724  - 0716
 0732  - 0724
 0740  - 0732
 0748  - 0740
 0756  - 0748
 0764  - 0756
 0824  - 1164
 0860  - 0860
 0864  - 1012 1028 1056 1080
 0876  - 0868 0884 0940
 0892  - 0876
 0916  - 0876
 0936  - 0916 0924 0976
 0956  - 0928 0932
 1000  - 0988
 1008  - 1000
 1028  - 0956
 1056  - 1044
 1072  - 0956
 1096  - 0956
 1112  - 1104
 1124  - 1096
 1132  - 1116
 1136  - 0908
 1156  - 1148
 1168  - 1140
 1172  - 1136
 1268  - 1324
 1292  - 1284
 1308  - 1300
 1332  - 1304
 1352  - 1224
 1356  - 1228
 1360  - 1232
 1364  - 1236
 1368  - 1240
 1372  - 1244
 1376  - 1248
 1380  - 1252
 1384  - 1256
 1388  - 1260
(1500) - 0728
(4000) - 0736
(6000) - 0744
(7000) - 0752
(9000) - 0760
```

```
D0     - 0232
D1     - 0232
D2     - 0232
D3     - 0232
D4     - 0232
D5     - 0232
D6     - 0232
D7     - 0232
D8     - 0232
D9     - 0232
E1     - 0236
E1$    - 0236 0428 0432 0436 0456 0460 0468 0488 0492 0516
E2$    - 0236 0428 0468 0516
H8$(   - 0240
K1     - 0584 0608 0632
K2     - 0584 0596 0620 0636
K3     - 0584 0588 0600
K4     - 0584 0592 0596 0624
K5     - 0600 0604 0628
K5$(   - 0240 0624
L1$    - 0684 0704
L2$    - 0684 0712 0720 0728 0736 0744 0752 0760
L3$    - 0684 0716 0724 0732 0740 0748 0756
N1$    - 0236 1276
N2$    - 0236
Q0(    - 0276
Q0$(   - 0276 1232 1236 1240 1244 1248 1252 1256 1260 1336
Q1$(   - 0252 1296
Q2$(   - 0288
Q3$(   - 0276
Q4$(   - 0280
Q5$(   - 0252
Q6$(   - 0280
Q7$(   - 0280
Q8$(   - 0284
Q9$(   - 0284
R      - 0316
R0     - 0244 1212 1280 1284 1288 1292 1296 1332 1336
R0$    - 0248
R0$(   - 0260 0860 0876 0916 0920 0924 0928 0932 0956 0980 1004 1008
R1     - 0244 0820 0828 0836 0976 1132 1144 1148 1152
R1$    - 0248 0820 0868 1000 1104 1136
R1$(   - 0284
R2     - 0244 1144 1152 1156 1168
R2$    - 0248 0852 0864 0896 0904 0980 1040 1072 1076 1100 1124 1128 1140 1168
         1212 1224 1280 1292 1296
R2$(   - 0260 0316 1296 1300 1304 1332
R3     - 0244 0848 0900 0904 0976 0980 0984 0988 1028 1036 1040 1096 1124 1132
         1148
R3$    - 0248
R4     - 0244
R4$    - 0252
R4$(   - 0260 0920
R5     - 0244
R5$    - 0252
R5$(   - 0260
R6     - 0244
R6$    - 0256 0820 1100 1108 1128 1132
```

```
R6$(   - 0264 0320 0608 0616 0632
R7     - 0244
R7$    - 0256
R7$(   - 0264 0324 0620 0636
R8     - 0244
R8$    - 0256
R8$(   - 0264 0448 0504 0608 0616 0620 0624 0632 0636 0828 0836 0840 0864 0872
         0880 0936 1156 1160 1316
R9     - 0244 0844 0864 0992 1044 1048 1072 1076
R9$    - 0256
R9$(   - 0264 0824 0828 0832 0836 1156
S0$    - 0268
S1$    - 0268
S2$    - 0268
S3$    - 0268
S4$    - 0272
S5$    - 0272
S6$    - 0272
S7$    - 0272
```

```
0424 DEFFN' 0
       - 0764
0344 DEFFN' 1
0352 DEFFN' 2
0360 DEFFN' 3
0584 DEFFN' 101
       - 0440 0464 0476 0496 0512 0524 0700 0708 1268 1308 1320
0684 DEFFN' 102
       - 0348 0356 0364
0820 DEFFN' 104
       - 1276
1212 DEFFN' 105
```

4.2.2 ATH2/II

This section of the 'AUTHOR/2' program contains subroutines 106 to 123.

The remainder of this section presents the 'ATH2/II' program listing and reference maps.

```
1500  %## Program > AUTHOR/2  (ATH2/II)   Version > 2.0  2200 T/VP   780320
1504  REM **  Subroutine  ********************************************** 1 0 6
1508  REM *   Abstract   >
1512  REM *     To accept and store mutiple lines of data.
1516  REM *     Each line being 63 characters and a buffer
1520  REM *     containing 32 lines.
1524  REM *   Variables  >
1528  REM *     I1  > id for data entered !
1532  REM *     I2  > storage functn code ! R2$ > scratch bufr
1536  REM ************************************************
1540  DEFFN ' 106 (I1,I2)                                  : REM ENTRY
1544  GOSUB ' 107                                          : REM INITIALIZATION
1548  GOSUB ' 108 (RO$(1))                                 : REM ENTER A CHARACTER
1552  GOSUB ' 109 (RO$(1))                                 : REM PROCESS CHARACTER
1556  IF          RO$(1) <> HEX(FF)          THEN 1548     : REM NOT END OF DATA
1560  RETURN                                               : REM EXIT
```

```
1564 REM **
1568 REM **
1572 REM **  Subroutine  ********************************************* 1 0 7
1576 REM **  Function >  To Initialize pointers and buffer
1580 REM **  Variables >
1584 REM **    R1    > crt pos <row>        ! R2    > crt pos <col>
1588 REM **    R3    > linear bufr indx     ! R6    > top lin on crt
1592 REM **    R4    > bufr pos <row>       ! R5    > bufr pos <col>
1596 REM **    R1$()> storage buffer        ! R0$()> charactr bufr
1600 REM **    R9$()> scratch               ! R9    > scratch
1604 REM **    R5$()> search id's           ! R8$()> micro cmnds
1608 REM **    E1 > pre-existing data flg !
1612 REM **
1616 REM *********************************************
1620  DEFFN ' 107                                          : REM ENTRY
1624              R1 = 1                                   : REM INIT CRT ROW POS
1628              R2 = 1                                   : REM INIT CRT COL POS
1632              R3 = 1                                   : REM INIT LINR BUF POS
1636              R4 = 1                                   : REM INIT BUFR ROW POS
1640              R5 = 1                                   : REM INIT BUFR COL POS
1644              R6 = 1                                   : REM INIT TOP OF SCREN
1648              E1 = 1                                   : REM CLR EXSTNG DAT FLG
1652  INIT(20)    R1$                                      : REM CLR CUSR RPEAT FLG
1656  INIT(20)    R2$                                      : REM CLR NUMBER BUFR
1660  INIT(20)    R1$()                                    : REM CLR TEXT BUFR
1664  INIT(20)    R9$()                                    : REM CLR SCRATCH BUFR
1668  INIT(20)    R0$()                                    : REM CLR ENTRY BUFR
1672  KEYIN       R0$(1), 1672, 1672                       : REM CLR ENTRY QUEUE
1676 REM **
1680  $GID HOME   /005 (4001,R8$(9))                       : REM HOME CUSOR
1684  FOR         R9 = 1 TO 13                             : REM BEG BORDER LOOP
1688  $GID BORDER /005 (R8$(2),R8$(9)) R9$()<1,63>         : REM PRINT BORDER
1692  NEXT        R9                                       : REM CONT LOOP
1696 REM **
1700  GOSUB ' 101 (14,50,1,12)                             : REM POSITION CUSOR
1704  PRINT       " Entry Mode ";
1708  KEYIN       R0$(1), 1708, 1708                       : REM CLR ENTRY BUFR
1712  GOSUB ' 101 (R1,R2,0,0)                              : REM POSITION CUSOR
1716  RETURN                                               : REM EXIT
```

```
1720 REM **
1724 REM **
1728 REM **  Subroutine  ********************************************** 1 0 8
1732  REM **    Function >  To Input a character and verify
1736  REM **                for validity.
1740  REM **    Variables>
1744  REM **      R0$() > character bufr ! R4 > bufr indx <row>
1748  REM **      R8$() > micro cmnds    ! R5 > bufr indx <col>
1752  REM **      R4$() > translations   !
1756  REM **
1760 REM ***********************************************
1764  DEFFN ' 108 (R0$(1))                                  : REM ENTRY
1768              STR(R8$(9),5,1) = R1$(R4,R5)              : REM STOR CURNT CHAR
1772  KEYIN       R0$(1), 1788, 1812                        : REM INPUT CHARACTER
1776  $GIO WAIT   /005(R8$(1),R8$(9))                       : REM INPUT WAIT
1780  GOTO        1772                                      : REM CONT WAIT
1784 REM **
1788  IF          R0$(1) < HEX(08)          THEN 1812       : REM INVALID INPUT
1792  $TRAN       (R0$() <1,1>, R4$()<257,16>) R            : REM CONV CTL CHARS
1796  IF          R0$(1)  > HEX(7F)         THEN 1812       : REM INVALID INPUT
1800  IF          R0$(1)  > HEX(1F)         THEN 1824       : REM VALID INPUT
1804  IF          R0$(1) <  HEX(06)         THEN 1824       : REM VALID INPUT
1808 REM **
1812  $GIO TONE   /005(R8$(7),R8$(9))                       : REM SOUND TONE
1816  GOTO        1772                                      : REM CONTINUE INPUT
1820 REM **
1824  RETURN                                                : REM EXIT
```

```
1828 REM **
1832 REM **
1836  REM **  Subroutine  ********************************************** 1 0 9
1840  REM **    Function >  To Insert Character into buffer
1844  REM **                  and process any control functions.
1848  REM **    Variables>
1852  REM **      RO$() > character buffer !
1856  REM **
1860  REM ********************************************
1864  DEFFN ' 109 (RO$(1))                                  : REM ENTRY
1868              RO = VAL(RO$(1))                          : REM CONV TO DEC
1872 REM **
1876 REM **       ! ERASE ! UNDRSCR ! EDIT ! ESCAPE ! CRLF ! OTHER !
1880 REM **       !-------!---------!------!--------!------!-------!
1884  ON RO GOTO    1900  ,  1904   , 1908 ,  1912  , 1916 , 1896  : REM BRNCH
1888 REM **       !-------!---------!------!--------!------!-------!
1892 REM **
1896  GOSUB ' 110 (RO$(1))                  : GOTO 1920 : REM ENTER CHARACTER
1900  GOSUB ' 111                           : GOTO 1920 : REM RUBOUT CHARACTER
1904  GOSUB ' 112                           : GOTO 1920 : REM UNDERSCORE CHAR.
1908  GOSUB ' 113                           : GOTO 1920 : REM EDIT BUFFER
1912  GOSUB ' 123                           : GOTO 1920 : REM INPUT COMPLETE
1916  GOSUB ' 124                           : GOTO 1920 : REM INSERT <CRLF>
1920  RETURN                                            : REM EXIT
```

```
1924 REM **
1928 REM **
1932  REM **  Subroutine  ********************************************* 1 1 0
1936  REM **  Function >  To transfer character from character
1940  REM **              buffer to storage buffer.
1944  REM **              Store buffer on disk when full.
1948  REM **  Variables>
1952  REM **    R1     > crt pos <row>  ! R2     > crt pos <col>
1956  REM **    R4     > buf pos <row>  ! R5     > buf pos <col>
1960  REM **    R3     > bufr pointer   ! R6     > top lin of buf
1964  REM **    R1$()  > storage bufr   ! R0$()  > char. buffer
1968  REM **    R5$()  > terminator char! I1     > item id
1972  REM **    I2     > storg fnct code!
1976  REM **
1980  REM ***********************************************
1984  DEFFN ' 110 (R0$(1))                                    : REM ENTRY
1988  IF          R3 =  1                          THEN 2168 : REM FIRST CHARACTER
1992 REM **
1996  IF          R2 <> 59                         THEN 2036 : REM NOT TIME FOR TONE
2000  $GIO TONE   /005 (R8$(7),R8$(9))                       : REM END OF LINE TONE
2004  IF          R4 <> 32                         THEN 2036 : REM NOT END OF BUFR
2008  GOSUB ' 101 (14,50,1,12)                              : REM POSITION CUSOR
2012  PRINT       " LAST LINE! "                              : REM PRINT WARNING
2016  $GIO TONE   /005 (R8$(4),R8$(9))                       : REM PAUSE TONE
2020  $GIO TONE   /005 (R8$(4),R8$(9))                       : REM PAUSE TONE
2024  $GIO TONE   /005 (R8$(4),R8$(9))                       : REM PAUSE TONE
2028  GOSUB ' 101 (R1,R2,0,0)                               : REM REPOSITION CUSOR
2032 REM **
2036  IF          R2 <> 63                         THEN 2148 : REM NOT END OF LINE
2040              R2 =  0                                   : REM SET CRT POS
2044              R5 =  0                                   : REM SET BUFFR POS
2048  IF          R1 <> 13                         THEN 2068 : REM NOT BOTM OF SCRN
2052              R6 =  R6 + 1                              : REM INCR TOP LIN INDX
2056  GOSUB ' 121 (R6)                                      : REM ROLL CRT SCREEN
2060              R1 =  12                                  : REM SET CRT POS INDX
2064 REM **
2068              R1 =  R1 +  1                             : REM INCR CRT ROW INDX
2072  GOSUB ' 101 (R1,R2+1,0,0)                             : REM POSITION CUSOR
2076  IF          R4 <> 32                         THEN 2144 : REM NOT END OF BUFR
2080  MAT COPY    R5$()<2,1> TO R1$()<R3-1,1>               : REM SET EOI BYTE
2084  GOSUB ' 101 (14,50,0,0)                               : REM POSITION CUSOR
2088  PRINT       " Working... "
2092  GOSUB ' 129 (I1,I2)                                   : REM WRITE TO DISK
2096              I2    = 2                                 : REM SET FUNCTION INDX
2100              R3    = 1                                 : REM SET BUFR INDEX
2104              R6    = 1                                 : REM SET TOP OF BUFFR
2108              R4    = 0  :  R5    = 0                   : REM SET BUF POS INDX
2112              R1    = 1  :  R2    = 0                   : REM SET CRT POS INDX
2116  GOSUB ' 122 (E1)                                      : REM RESET TEXT BUFR
2120  GOSUB ' 101 (14,1,0,0)                                : REM POSITION CUSOR
2124  INIT(2D)    R9$()                                     : REM CREATE BOTOM LINE
2128  $GIO PRINT  /005(R8$(2),R8$(9)) R9$()<1,63>           : REM PRINT BOTOM LINE
2132  GOSUB ' 101 (14,50,0,0)                               : REM POSITION CUSOR
2136  PRINT       " Entry Mode "
2140  GOSUB ' 101 (R1,R2+1,0,0)                             : REM POSITION CUSOR
2144              R4 =  R4 + 1                              : REM INCR ROW INDX BUF
2148              R2 =  R2 + 1                              : REM INCR COL INDX CRT
2152              R5 =  R5 + 1                              : REM INCR COL INDX BUF
```

```
2156  $GIO LEFT   /005 (4009     ,R8$(9))                          : REM SPACE CUSOR LEFT
2160  IF          R2 <> 1                              THEN 2168 : REM SPACE IS OK
2164  $GIO RIGHT  /005  (4008     ,R8$(9))                        : REM SPACE CUSOR RIGHT
2168  MAT   COPY  RO$() <1,1> TO R1$() <R3,1>                     : REM LOAD INTO BUFFER
2172  $GIO  CHAR  /005  (A0004008,R8$(9)) RO$() <1,1>             : REM DISPLAY CHARACTER
2176              R3  = R3 +  1                                   : REM INCR BUFFER INDX
2180  RETURN                                                      : REM EXIT
```

```
2184 REM **
2188 REM **
2192  REM **  Subroutine  ********************************************** 1 1 1
2196  REM **  Function > To Rubout current character.
2200  REM **  Variables >
2204  REM **    R1    > crt pos <row>  ! R2    > crt pos <col>
2208  REM **    R4    > buf pos <row>  ! R5    > buf pos <col>
2212  REM **    R3    > bufr pointer   ! R6    > top lin of buf
2216  REM **    R1$() > storage bufr   ! R0$() > char. buffer
2220  REM **
2224  REM **********************************************
2228  DEFFN ' 111                                              : REM ENTRY
2232  IF          R3 =  1                          THEN 2300 : REM IF BUFR BEGINING
2236  GOSUB ' 101 (R1,R2,1,1)                               : REM ERASE CHARACTER
2240  MAT COPY    R5$() <1,1> TO R1$() <R3-1,1>             : REM CLR TEXT BUFFR
2244              R3 =  R3 -  1                             : REM DECRMNT BUFR INDX
2248  IF          R3 =  1                          THEN 2300 : REM IF BUFR BEGINING
2252  IF          R2 <> 1                          THEN 2288 : REM NOT BEG OF LINE
2256              R2 =  64                                  : REM SET TO END OF LINE
2260              R5 =  64                                  : REM SET TO END OF LINE
2264  IF          R1 <> 1                          THEN 2280 : REM NOT TOP OF SCREEN
2268              R6 =  R6 - 1                              : REM DECRMNT TOP INDEX
2272  GOSUB ' 121 (R6)                                      : REM RE-WRITE SCREN PAC
2276              R1 =  2                                   : REM RESET CRT ROW INDX
2280              R1 =  R1 -  1                             : REM DECRMNT ROW INDX
2284              R4 =  R4 -  1                             : REM DECRMNT ROW INDX
2288              R2 =  R2 -  1                             : REM DECRMNT COL INDX
2292              R5 =  R5 -  1                             : REM DECRMNT COL INDX
2296  GOSUB ' 101 (R1,R2,0,0)                               : REM POSITION CUSOR
2300  RETURN                                                : REM EXIT
```

```
2304 REM **
2308 REM **
2312  REM **  Subroutine  ********************************************** 1 1 2
2316  REM **  Function > To underscore the current character.
2320  REM **  Variables >
2324  REM **    R1$() > storage bufr    ! RO$() > char buffer
2328  REM **    R3    > buffer pointer !
2332  REM **
2336  REM ***********************************************
2340  DEFFN ' 112                                          : REM ENTRY
2344  MAT COPY    R1$() <R3-1,1> TO RO$() <1,1>            : REM FETCH CHARACTER
2348  AND         (RO$(1),7F)                              : REM CLR UNDRSCR FLAG
2352  ADD         (RO$(1),80)                              : REM SET UNDRSCR FLAG
2356  MAT COPY    RO$() <1,1>    TO R1$() <R3-1,1>         : REM SAVE CHARACTER
2360  RETURN                                               : REM EXIT
```

```
2364 REM **
2368 REM **
2372  REM **  Subroutine  ********************************************** 1 1 3
2376  REM **  Function > To alow the storage buffer to be
2380  REM **               Edited. Editing can only be performed
2384  REM **               on data currently in buffer.
2388  REM **  Variables >
2392  REM **    IO   > value of numbr str ! R3  > buffer pointer
2396  REM **    R2$  > number string      ! R4  > buf indx <row>
2400  REM **    R1$()> storage buffer     ! R5  > buf indx <col>
2404  REM **    RO$()> character bufr     ! R4$()> translations
2408  REM **    R8$()> micro cmnds        ! I9  > length of numbr string
2412  REM **
2416  REM ***********************************************
2420  DEFFN ' 113                                        : REM ENTRY
2424  GOSUB ' 101 (14,50,1,12)                           : REM POSITION CUSOR
2428  PRINT       " Edit  Mode "
2432  GOSUB ' 101 (R1,R2,0,0)                            : REM POSITION CUSOR
2436 REM **
2440  INIT(20)    R2$                                    : REM CLR STORAGE BUFR
2444              I9 = 0                                 : REM SET NUMBR LENGTH
2448              IO = 1                                 : REM SET NUMBR VALUE
2452              R3 = (R4-1)*63 + R5 + 1                : REM CMPUTE CURNT INDX
2456  IF          R1$ = HEX(00)            THEN 2544     : REM REPEAT CUSR MOVE
2460 REM **
2464              STR(R8$(9),5,1) = R1$(R4,R5)           : REM STOR CURNT CHAR
2468  KEYIN       RO$(1), 2484, 2584                     : REM INPUT EDIT CMND
2472  $GIO CUSOR  /005 (R8$(1),R8$(9))                   : REM INPUT WAIT
2476  GOTO        2468                                   : REM CONTINUE WAIT
2480 REM **
2484  $TRAN       (RO$()<1,1>, R4$()<273,48>) R          : REM CONV CTL CHAR
2488              R1$ = HEX(20)                          : REM CLR REPT CUSR FLG
2492              RO = VAL(RO$(1))                       : REM CMPUT CHAR VALUE
2496  IF          RO$(1) <  HEX(0A)        THEN 2608     : REM VALID CTL CHAR
2500  IF          RO$(1)  > HEX(39)        THEN 2584     : REM INVALID INPUT
2504  IF          RO$(1)  > HEX(29)        THEN 2608     : REM VALID NUM CHAR
2508  IF          RO$(1)  > HEX(0A)        THEN 2536     : REM INVALID INPUT
2512  IF          R3      <> 2             THEN 2564     : REM NO TEXT LOOKUP
2516  IF          E1      <> 1             THEN 2564     : REM IF NOT A NEW START
2520  GOSUB ' 122 (2)                                    : REM LOAD TEXT BUFFER
2524  IF          I4 = 0                   THEN 2564     : REM NON-EXISTANT ITEM
2528  GOTO        2440                                   : REM BRANCH EDIT INPUT
2532 REM **
2536              R1$ = HEX(00)                          : REM SET REPT CUSR FLG
2540              RO  = VAL(RO$(1)) - 10                 : REM CMPUT BRANCH INDX
2544  KEYIN       RO$(1), 2484, 2544                     : REM REPEAT INTRUP CHK
2548              STR(R8$(9),5,1) = R1$(R4,R5)           : REM STOR CURNT CHAR
2552  $GIO CUSOR  /005 (R8$(1),R8$(9))                   : REM DISPLAY CUSOR
2556  GOTO        2608                                   : REM EXCUTE MOVE
2560 REM **
2564  GOSUB ' 101 (14,50,1,12)                           : REM POSITION CUSOR
2568  PRINT       " Entry Mode "
2572  GOSUB ' 101 (R1,R2,0,0)                            : REM POSITION CUSOR
2576  GOTO        2668                                   : REM BRANCH TO EXIT
2580 REM **
2584  $GIO TONE   /005 (R8$(7),R8$(9))                   : REM ERROR TONE
2588  GOTO        2440                                   : REM BRANCH EDIT INPUT
```

```
2592 REM **
2596 REM **
2600 REM **        !  LEFT  ! RIGHT !   UP   !  DOWN  ! CHANG !
2604 REM **        !--------!-------!--------!--------!-------!
2608  ON RO GOTO     2628   , 2632  ,  2636  ,  2640  , 2644  ,
                     2648   , 2652  ,  2656  ,  2660  , 2624  : REM BRANCH
2612 REM **        !--------!-------!--------!--------!-------!
2616 REM **        ! DELET  ! INSRT ! BEGIN  !  END   ! NUMBR !
2620 REM **
2624  GOSUB ' 120 (IO,I9)                    : GOTO 2464 : REM CMPUT NUMBER
2628  GOSUB ' 114 (IO*(-1))                  : GOTO 2440 : REM CUSOR LEFT
2632  GOSUB ' 114 (IO)                       : GOTO 2440 : REM CUSOR RIGHT
2636  GOSUB ' 115 (IO*(-1))                  : GOTO 2440 : REM CUSOR UP
2640  GOSUB ' 115 (IO)                       : GOTO 2440 : REM CUSOR DOWN
2644  GOSUB ' 116                            : GOTO 2440 : REM CHANGE CHARACTER
2648  GOSUB ' 117                            : GOTO 2440 : REM DELETE CHARACTER
2652  GOSUB ' 118 (IO)                       : GOTO 2440 : REM INSERT CHARACTER
2656  GOSUB ' 119 (1)                        : GOTO 2440 : REM SET TO BEG OF BFR
2660  GOSUB ' 119 (2)                        : GOTO 2440 : REM SET TO END OF BFR
2664 REM **
2668  RETURN                                               : REM EXIT
```

```
2672  REM **
2676  REM **
2680  REM **  Subroutine  ********************************************** 1 1 4
2684  REM **  Function > To position cusor left or right of
2688  REM **               current position.
2692  REM **  Variables >
2696  REM **    RO$() > character bufr ! R1 > crt pos <row>
2700  REM **    R5    > buf indx <col> ! R2 > crt pos <col>
2704  REM **    R3    > bufr indx linr ! R4 > buf indx <row>
2708  REM **    IO    > length of move !
2712  REM **
2716  REM ***********************************************
2720  DEFFN ' 114 (IO)                                    : REM ENTRY
2724              R2 = R2 + IO                            : REM CMPUT NEW COL INDX
2728  IF            R2 > 0                  THEN 2744     : REM NOT PAST LEFT MRGN
2732              R2 = R2 + 63                            : REM SCLE FOR RAPAROUND
2736  GOTO          2752                                  : REM BRANCH FOR SCROLL
2740 REM **
2744  IF            R2 < 64                 THEN 2752     : REM NOT PAST RGHT MRGN
2748              R2 = R2 - 63                            : REM SCLE FOR RAPAROUND
2752              R5 = R2                                 : REM SET NEW COL INDX
2756  GOSUB ' 101 (R1,R2,0,0)                             : REM POSITION CUSUR
2760  RETURN                                              : REM EXIT
```

```
2764 REM **
2768 REM **
2772  REM **  Subroutine  ********************************************** 1 1 5
2776  REM **  Function > To position cusor up or down of
2780  REM **             current position.
2784  REM **  Variables >
2788  REM **    RO$() > character bufr ! R1 > crt pos <row>
2792  REM **    R5    > buf indx <col> ! R2 > crt pos <col>
2796  REM **    R3    > bufr indx linr ! R4 > buf indx <row>
2800  REM **    IO    > length of move ! R1$()> storage bufr
2804  REM **    R8$() > micro cmnds    !
2808  REM **
2812  REM ***********************************************
2816  DEFFN ' 115 (IO)                                    : REM ENTRY
2820              R4 = R4 + IO                            : REM CMPUT NEW ROW IDX
2824  IF          R4 > 0                        THEN 2840 : REM NOT OVER TOP
2828              R4 = R4 + 32                            : REM SCLE FOR RAPAROUND
2832  GOTO        2852                                    : REM BRANCH FOR SCROLL
2836 REM **
2840  IF          R4 < 33                       THEN 2852 : REM NOT OVER BOTTOM
2844              R4 = R4 - 32                            : REM SCLE FOR RAPAROUND
2848 REM **
2852  IF          R4 >  R6+12                   THEN 2864 : REM IF PAST BOTTOM
2856  IF          R4 >= R6                      THEN 2884 : REM IF ON SCREEN
2860 REM **
2864              R6  = R4                                : REM SET TOP OF PAGE
2868  IF          IO <  0                       THEN 2880 : REM MOVE WAS CUSR UP
2872              R6  = R6 - 12                           : REM RESET TOP OF PAG
2876 REM **
2880  GOSUB ' 121 (R6)                                    : REM REWRITE SCREEN
2884              R1 = R4 - R6 + 1                        : REM RESET CRT POSITION
2888  GOSUB ' 101 (R1,R2,0,0)                           : REM POSITION CUSOR
2892  RETURN                                            : REM EXIT
```

```
2896 REM **
2900 REM **
2904  REM **  Subroutine  ********************************************** 1 1 6
2908  REM **  Function > To Change current character only.
2912  REM **
2916  REM **  Variables >
2920  REM **    RO$() > character buffer ! R1$() > storage bufr
2924  REM **    R3    > buffer indx      !
2928  REM **
2932  REM ***********************************************
2936  DEFFN ' 116                                          : REM ENTRY
2940  GOSUB ' 108 (RO$(1))                                 : REM ENTER A CHARACTER
2944              RO = VAL(RO$(1))                         : REM CONV TO AN INDX
2948 REM **
2952 REM **       ! ERASE ! UNDRSCR ! EDIT ! ESCAPE ! CRLF ! OTHER !
2956 REM **       !-------!---------!------!--------!------!-------!
2960  ON RO GOTO    2988  ,  2976   , 2992 , 2992   , 2988 , 2972  : REM BRNCH
2964 REM **       !-------!---------!------!--------!------!-------!
2968 REM **
2972  GOSUB ' 110 (RO$(1))                 :  GOTO 3000 : REM INSERT CHAR
2976  GOSUB ' 112                         :  GOTO 3000 : REM UNDERSCORE CHAR
2980  GOTO        3000                                   : REM BRANCH TO EXIT
2984 REM **
2988  $G10 TONE   /005 (R8$(7),R8$(9))                   : REM INVALID CHARACTER
2992              IO = 0                                 : REM SET END INDEX
2996 REM **
3000              IO = IO - 1                            : REM DECRMNT INDEX
3004  IF          IO > 0                      THEN 2940  : REM CONT CHAR CHANGE
3008  RETURN                                             : REM EXIT
```

```
3012 REM **
3016 REM **
3020  REM **  Subroutine  ********************************************** 1 1 7
3024  REM **  Function > To Delete a string of characters.
3028  REM **
3032  REM **  Variables >
3036  REM **    R1$() > storage buffer ! R0  > spcs in bufr
3040  REM **    R9$() > scratch buffer ! I0  > spcs to delete
3044  REM **    R3    > buffer index   ! R1  > crt pos <row>
3048  REM **    R2    > crt pos <col>  ! R6  > top line of buf
3052  REM **
3056  REM ***********************************************
3060  DEFFN ' 117                                          : REM ENTRY
3064              R0 = 2016 - R3 + 1                       : REM CMPUT SPCE AVIABLE
3068  IF          I0 < R0                      THEN 3076   : REM SPACE AVAIABLE
3072              I0 = R0                                  : REM SET TO MAX SPACE
3076              I9 = 2016 - R3 + 1                       : REM CMPUT TRNSFR SPACE
3080  MAT COPY    R1$()<R3+I0,I9-I0> TO R1$()<R3,I9>       : REM CLR BUFR TEXT
3084  GOSUB ' 121 (R6)                                     : REM RE-WRITE SCREEN
3088  GOSUB ' 101 (R1,R2,0,0)                              : REM POSITION CUSOR
3092  RETURN                                               : REM EXIT
```

```
3096 REM **
3100 REM **
3104  REM **  Subroutine  ********************************************** 1 1 8
3108  REM **  Function > To Insert characters into the buffer.
3112  REM **
3116  REM **  Variables >
3120  REM **    R0    > max spcs to insrt! R1  > crt pos <row>
3124  REM **    R2    > crt pos <col>    ! R3  > buffer indx
3128  REM **    R1$() > storage buffer   ! R4  > buf pos <row>
3132  REM **    R2$() > search result    ! R5  > buf pos <col>
3136  REM **    R9$() > scratch buffer   ! I0  > spcs to insrt
3140  REM **
3144  REM ***********************************************
3148  DEFFN ' 118 (I0)                                      : REM ENTRY
3152  GOSUB ' 136 (32)                                      : REM DETERMIN BUFR LEN
3156              R0 = 2016 - R0                            : REM CMPUT CHAR REMAING
3160  IF          I0 < R0                        THEN 3168  : REM SPACE IS AVAIABLE
3164              I0 = R0 - 1                               : REM SET TO MAX SPACE
3168              I9 = (2016-R0) - R3 + 2                   : REM CMPUT TRNSFR SPACE
3172  MAT COPY    -R1$()<R3,I9> TO -R1$()<R3+I0,I9>         : REM REVERSE COPY
3176  MAT COPY    R5$()<1,1> TO R1$()<R3,I0>                : REM CLR INSRT SPAC
3180  GOSUB ' 121 (R6)                                      : REM RE-WRITE SCREEN
3184  GOSUB ' 101 (14,50,1,12)                              : REM POSITION CUSOR
3188  PRINT       " Insert Mode "
3192  GOSUB ' 101 (R1,R2,0,0)                               : REM POSITION CUSOR
3196  GOSUB ' 108 (R0$(1))                                  : REM ENTER A CHARACTER
3200              R0 = VAL(R0$(1))                          : REM CONV TO AN INDX
3204 REM **
3208 REM **       ! ERASE ! UNDRSCR ! EDIT ! ESCAPE ! CRLF ! OTHER !
3212 REM **       !-------!---------!------!--------!------!-------!
3216  ON R0 GOTO    3232  ,  3236   , 3240 ,  3240  , 3240 , 3228  : REM BRNCH
3220 REM **       !-------!---------!------!--------!------!-------!
3224 REM **
3228  GOSUB ' 110 (R0$(1))                 : GOTO 3196 : REM SAVE CHAR IN BUFR
3232  GOSUB ' 111                          : GOTO 3196 : REM RUBOUT CHARACTER
3236  GOSUB ' 112                          : GOTO 3196 : REM UNDERSCORE CHAR.
3240  $GIO TONE   /005 (R8$(7),R8$(9))                 : REM EXIT INSERT MODE
3244  GOSUB ' 101 (14,50,1,12)                          : REM POSITION CUSOR
3248  PRINT       " Edit Mode "
3252  GOSUB ' 101 (R1,R2,0,0)                           : REM POSITION CUSOR
3256  RETURN                                            : REM EXIT
```

```
3260 REM **
3264 REM **
3268 REM **  Subroutine  ********************************************** 1 1 9
3272 REM **  Abstract >
3276 REM **    To position the cusor at the begining or end of the
3280 REM **    text buffer.
3284 REM **  Variables >
3288 REM **    I9 > position index          ! R6 > top lin on crt
3292 REM **    R1   > crt pos <row>         ! R2   > crt pos <col>
3296 REM **    R4   > bufr pos <row>        ! R5   > bufr pos <col>
3300 REM **
3304 REM ************************************************
3308  DEFFN ' 119 (I9)                                    : REM ENTRY
3310  REM
3312  REM          ! TOP OF BUFFER ! END OF BUFFER !
3314  REM          !---------------!---------------!
3316  ON I9 GOTO       3322        ,     3324          : REM BRANCH
3318  REM          !---------------!---------------!
3320  REM
3322               R1= 1: R2=1: R4= 1: R5=1: R6= 1     : GOTO 3328
3324               R1=13: R2=1: R4=32: R5=1: R6=20     : GOTO 3328
3326 REM **
3328  GOSUB ' 121 (R6)                                  : REM REWRITE SCREEN
3330  GOSUB ' 101 (R1,R2,0,0)                           : REM POSITION CUSOR
3332  RETURN                                            : REM EXIT
```

```
3340 REM **
3344 REM **
3348  REM **  Subroutine  ********************************************** 1 2 0
3352  REM **  Function > To compute the cycle number.
3356  REM **  Variables >
3360  REM **    R2$ > number string  ! I0 > length of string
3364  REM **
3368  REM ***********************************************
3372  DEFFN ' 120 (I0,I9)                                    : REM ENTRY
3376              I9 = I9 + 1                                : REM INCR NUMBR INDX
3380              STR(R2$,I9,1) = R0$(1)                     : REM INSRT DIGIT
3384  IF          NUM(R2$) <> 64            THEN 3404        : REM INVALID NUMBER
3388  CONVERT     R2$ TO I0                                  : REM CMPUT NUMRC VALUE
3392  IF          I0 > 64                   THEN 3404        : REM ILLEAGL VALUE
3396  IF          I0 > 0                    THEN 3420        : REM LEAGL VALUE
3400 REM **
3404  $GIO TONE   /005 (R8$(7),R8$(9))                       : REM INPUT ERROR
3408              I0 =  1                                    : REM SET TO DEFALT VAL
3412              I9 =  0                                    : REM RESET NUMBR LEN
3416  INIT(20)    R2$                                        : REM CLR NUMBR BUFR
3420  RETURN                                                 : REM EXIT
```

```
3424 REM **
3428 REM **
3432  REM **  Subroutine  ********************************************** 1 2 1
3436  REM **  Function >  To roll screen when position is
3440  REM **              advanced off of screen.
3444  REM **  Variables >
3448  REM **    R9     > scratch       ! R1$() > storage bufr
3452  REM **    R9$() > scratch        ! R6    > top of pag indx
3456  REM **
3460  REM **********************************************
3464  DEFFN ' 121 (R6)                                    : REM ENTRY
3468  INIT(2E)    R9$()                                   : REM LOAD SCRATCH BUFR
3472  $GIO HOME   /005 (4001,R8$(9))                      : REM HOME CUSOR
3476  REM **
3480  FOR         R9 = R6 TO R6+12                        : REM BEG REWRITE LOOP
3484   IF         R9 <= 0                      THEN 3492  : REM OUTSIDE PAG LIMIT
3488   IF         R9 <= 32                     THEN 3500  : REM INSIDE PAG LIMITS
3492   $GIO FILL  /005 (R8$(2),R8$(9)) R9$()<1,63>        : REM PRINT FILL LINE
3496   GOTO       3504                                    : REM CONT OUTPUT LOOP
3500   $GIO PRINT /005 (R8$(2),R8$(9)) R1$()<(R9-1)*63+1,63> : REM PRINT TEXT
3504  NEXT        R9                                      : REM CONT REWRITE LOOP
3508  REM **
3512  RETURN                                              : REM EXIT
```

```
3516 REM **
3520 REM **
3524 REM **  Subroutine  ********************************************** 1 2 2
3528 REM **  Abstract >
3532 REM **    To load storage buffer with text before editing.
3536 REM **  Variables >
3540 REM **    E1 > text recall flag       ! I4 > bufr memory position
3544 REM **    R6 > line at top of screen  ! D0 > memory bufr subscript
3548 REM **    D1 > index sector adress    ! D2 > subsector index
3552 REM **    D9 > scratch adrs register  ! Q6$() > memory buffer
3556 REM **    Q8$() index sector bufr    !
3560 REM ***********************************************
3564  DEFFN ' 122 (E1)                                    : REM ENTRY
3568  IF          E1 =  4                  THEN 3576    : REM IF END OF DATA
3572  IF          E1 <> 1                  THEN 3592    : REM NOT INITIAL ENTRY
3576  INIT (20)   R1$()                                  : REM CLR TEXT BUFR
3580  GOSUB ' 121 (R6)                                    : REM REWRITE SCREEN
3584  GOTO        3652                                    : REM BRANCH TO EXIT
3588  REM **
3592  IF          E1 <> 2                  THEN 3628    : REM NOT STRT OF EDIT
3596  GOSUB ' 130 (I1)                                    : REM DISK LOOKUP
3600  IF          I4 = 0                   THEN 3652    : REM NON-EXISTANT ITEM
3604              R0$ = Q8$(D2+1)                         : REM STOR BEG ADDRESS
3608              Q6$(D0,D5) = HEX(0000)                  : REM CLR MEMORY TBL
3612              Q8$(D2) = HEX(0000)                     : REM CLR DISK INDEX
3616              Q8$(D2+1) = HEX(FFFF)                   : REM CLR DISK INDEX
3620  DATA SAVE   BA T #2, (D1,D9) Q8$()                  : REM UPDATE DISK INDEX
3624 REM **
3628              S0$ = R0$                               : REM STOR BEG SCRT ADRS
3632  GOSUB ' 125 (R0$,R0,I1)                             : REM LOAD STORAGE BUFR
3636  GOSUB ' 135 (S0$,HEX(0000))                         : REM SRCH FOR NXT SCTR
3640              R6 = 1                                  : REM SET TOP OF PAGE
3644  GOSUB ' 121 (R6)                                    : REM REWRITE SCREEN
3648  GOSUB ' 101 (R1,R2,0,0)                             : REM POSITION CUSOR
3652  RETURN                                              : REM EXIT
```

```
3656 REM **
3660 REM **
3664  REM **  Subroutine  ********************************************** 1 2 3
3668  REM **  Function > To set End of Data Flag and
3672  REM **             Save buffer on disk.
3676  REM **  Variables >
3680  REM **    R2$  > eof flag          ! I1 > item id
3684  REM **    I2   > strge fnct code !
3688  REM **
3692  REM **********************************************
3696  DEFFN ' 123                                          : REM ENTRY
3700  GOSUB ' 136 (32)                                     : REM DETERMIN BUFR SIZ
3704  IF          R0 <> 2016                THEN 3712      : REM NOT END OF BUFR
3708              R0 =  R0 - 1                             : REM RESET END OF BUFR
3712  MAT COPY    R5$()<2,1> TO R1$()<R0+1,1>              : REM INSRT EOI CHAR
3716  GOSUB ' 129 (I1,I2)                                  : REM SAVE DATA ON DISK
3720              R0$(1) = HEX(FF)                         : REM SET EOF FLAG
3724  RETURN                                               : REM EXIT
```

1548 - 1556
1672 - 1672
1708 - 1708
1772 - 1780 1816
1788 - 1772
1812 - 1772 1788 1796
1824 - 1800 1804
1896 - 1884
1900 - 1884
1904 - 1884
1908 - 1884
1912 - 1884
1916 - 1884
1920 - 1896 1900 1904 1908 1912 1916
2036 - 1996 2004
2068 - 2048
2144 - 2076
2148 - 2036
2168 - 1988 2160
2280 - 2264
2288 - 2252
2300 - 2232 2248
2440 - 2528 2588 2628 2632 2636 2640 2644 2648 2652 2656 2660
2464 - 2624
2468 - 2476
2484 - 2468 2544
2536 - 2508
2544 - 2456 2544
2564 - 2512 2516 2524
2584 - 2468 2500
2608 - 2436 2504 2556
2624 - 2608
2628 - 2608
2632 - 2608
2636 - 2608
2640 - 2608
2644 - 2608
2648 - 2608
2652 - 2608
2656 - 2608
2660 - 2608
2668 - 2576
2744 - 2728
2752 - 2736 2744
2840 - 2824
2852 - 2832 2840
2864 - 2852
2880 - 2868
2884 - 2856
2940 - 3004
2972 - 2960
2976 - 2960
2988 - 2960
2992 - 2960
3000 - 2972 2976 2980
3076 - 3068
3168 - 3160
3196 - 3228 3232 3236

```
3228  - 3216
3232  - 3216
3236  - 3216
3240  - 3216
3404  - 3384 3392
3420  - 3396
3492  - 3484
3500  - 3488
3504  - 3496
3576  - 3568
3592  - 3572
3628  - 3592
3652  - 3584 3600
3712  - 3704
```

```
D0     - 3608
D1     - 3620
D2     - 3604 3612 3616
D5     - 3608
D9     - 3620
E1     - 1648 2116 2516 3564 3568 3572 3592
I0     - 2448 2624 2628 2632 2636 2640 2652 2720 2724 2816 2820 2868 2992 3000
         3004 3068 3072 3080 3148 3160 3164 3172 3176 3372 3388 3392 3396 3408
I1     - 1540 2092 3596 3632 3716
I2     - 1540 2092 2096 3716
I4     - 2524 3600
I9     - 2444 2624 3076 3080 3168 3172 3308 3320 3324 3372 3376 3380 3412
Q6$(   - 3608
Q8$(   - 3604 3612 3616 3620
R0     - 1868 1884 2492 2540 2608 2944 2960 3064 3068 3072 3156 3160 3164 3168
         3200 3216 3632 3704 3708 3712
R0$    - 3604 3628 3632
R0$(   - 1548 1552 1556 1668 1672 1708 1764 1772 1788 1792 1796 1800 1804 1864
         1868 1896 1984 2168 2172 2344 2348 2352 2356 2468 2484 2492 2496 2500
         2504 2508 2540 2544 2940 2944 2972 3196 3200 3228 3380 3720
R1     - 1624 1712 2028 2048 2060 2068 2072 2112 2140 2236 2264 2276 2280 2296
         2432 2572 2756 2884 2888 3088 3192 3252 3312 3332 3648
R1$    - 1652 2456 2488 2536
R1$(   - 1660 1768 2080 2168 2240 2344 2356 2464 2548 3080 3172 3176 3500 3576
         3712
R2     - 1628 1712 1996 2028 2036 2040 2072 2112 2140 2148 2160 2236 2252 2256
         2288 2296 2432 2572 2724 2728 2732 2744 2748 2752 2756 2888 3088 3192
         3252 3312 3332 3648
R2$    - 1656 2440 3380 3384 3388 3416
R3     - 1632 1988 2080 2100 2168 2176 2232 2240 2244 2248 2344 2356 2452 2512
         3064 3076 3080 3168 3172 3176
R4     - 1636 1768 2004 2076 2108 2144 2284 2452 2464 2548 2820 2824 2828 2840
         2844 2852 2856 2864 2884 3316 3324
R4$(   - 1792 2484
R5     - 1640 1768 2044 2108 2152 2260 2292 2452 2464 2548 2752 3316
R5$(   - 2080 2240 3176 3712
R6     - 1644 2052 2056 2104 2268 2272 2852 2856 2864 2872 2880 2884 3084 3180
         3320 3328 3464 3480 3580 3640 3644
R8$(   - 1680 1688 1768 1776 1812 2000 2016 2020 2024 2128 2156 2164 2172 2464
         2472 2548 2552 2584 2988 3240 3404 3472 3492 3500
R9     - 1684 1692 3480 3484 3488 3500 3504
R9$(   - 1664 1688 2124 2128 3468 3492
S0$    - 3628 3636
```

```
AUTHOR System: Phase 2 - ATH2/11   - ' MAP

???? DEFFN' 101
       - 1700 1712 2008 2028 2072 2084 2120 2132 2140 2236 2296 2424 2432 2564
         2572 2756 2888 3088 3184 3192 3244 3252 3332 3648
1540 DEFFN' 106
1620 DEFFN' 107
       - 1544
1764 DEFFN' 108
       - 1548 2940 3196
1864 DEFFN' 109
       - 1552
1984 DEFFN' 110
       - 1896 2972 3228
2228 DEFFN' 111
       - 1900 3232
2340 DEFFN' 112
       - 1904 2976 3236
2420 DEFFN' 113
       - 1908
2720 DEFFN' 114
       - 2628 2632
2816 DEFFN' 115
       - 2636 2640
2936 DEFFN' 116
       - 2644
3060 DEFFN' 117
       - 2648
3148 DEFFN' 118
       - 2652
3308 DEFFN' 119
       - 2656 2660
3372 DEFFN' 120
       - 2624
3464 DEFFN' 121
       - 2056 2272 2880 3084 3180 3328 3580 3644
3564 DEFFN' 122
       - 2116 2520
3696 DEFFN' 123
       - 1912
???? DEFFN' 124
       - 1916
???? DEFFN' 125
       - 3632
???? DEFFN' 129
       - 2092 3716
???? DEFFN' 130
       - 3596
???? DEFFN' 135
       - 3636
???? DEFFN' 136
       - 3152 3700
```

4.2.3 ATH3/II

This section of the 'AUTHOR/2' program contains subroutines 124 to 136.

The remainder of this section presents the 'ATH3/II' program listing and reference maps.

```
4000  %## Program > AUTHOR/2  (ATH3/II)   Version > 2.0  2200 T/VP   780320
4004  REM **  Subroutine  ********************************************** 1 2 4
4008  REM **  Function >  To position cusor and buffer pointer
4012  REM **                such that the next character entered will
4016  REM **                appear on next line.
4020  REM **  Variables >
4024  REM **    RO$() > character buffer     ! R2 > crt pos <col>
4028  REM **    R5 > buffer indx <col>       ! R4 > buffer indx <row>
4032  REM **    R1 > crt pos <row>           ! R3 > linear bufr indx
4036  REM **
4040  REM ***********************************************
4044  DEFFN ' 124                                          : REM ENTRY
4048              RO$(1) = HEX(20)                         : REM CLR INPUT BUFR
4052  IF          R2 = 63                       THEN 4076  : REM END OF LINE
4056  GOSUB ' 101 (R1,R2+1,1,63-R2)                        : REM ERASE SCRN TEXT
4060  MAT COPY    RO$()<1,1> TO R1$()<R3,63-R2>            : REM ERASE BUFR TEXT
4064              R2 = 62                                  : REM SET TO LINE END
4068              R5 = 62                                  : REM SET TO LINE END
4072 REM **
4076  IF          R4 <> 32                      THEN 4100  : REM NOT LAST LINE
4080  GOSUB ' 101 (14,50,1,12)                             : REM POSITION CUSOR
4084  PRINT       " LAST LINE! "                           : REM PRINT MESSAGE
4088  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM WARNING TONE
4092  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM WARNING TONE
4096  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM WARNING TONE
4100  GOSUB ' 101 (R1,R2,0,0)                              : REM POSITION CUSOR
4104              R3 = (R4-1)*63 + R5 + 1                  : REM CMPUT BUFR POS
4108  GOSUB ' 110 (RO$(1))                                 : REM SAVE CHAR IN BUFR
4112  RETURN                                               : REM EXIT
```

```
4116 REM **
4120 REM **
4124 REM **  Subroutine  ********************************************** 1 2 5
4128 REM **  Abstract >
4132 REM **    To load storage buffer from disk given begining
4136 REM **    sector address.
4140 REM **  Variables  >
4144 REM **     R0$    > begning sctr adrs   ! R0 > bufr indx
4148 REM **     R3$    > scrtch indx adrs    ! R2 > indx item flg
4152 REM **     G0$() > transfer buffer      ! G6() > bufr flag
4156 REM **     R1$() > storage buffer       ! R2$()> srch rsults
4160 REM **     R1     > item id             !
4164 REM **
4168 REM ***********************************************
4172  DEFFN ' 125 (R0$,R0,I1)                                  : REM ENTRY
4176  INIT(20)    R1$()                                        : REM CLR TEXT BUFFER
4180              R0 = 1                                       : REM SET BUFR INDX
4184  DATA LOAD   BA T #2, (R0$,R3$) G9$()                     : REM LOAD DATA
4188  MAT COPY    G9$()<1,252> TO R1$()<R0,252>                : REM LOAD BUFFR
4192  IF          G9$(64) <> HEX(FFFFFFFF)  THEN 4232          : REM NOT END OF BUFR
4196  MAT SEARCH  R1$(), = R5$(2) TO R2$()                     : REM SRCH FOR EOI
4200  IF          R2$(1) <> HEX(0000)       THEN 4260          : REM EOI FOUND
4204  GOSUB ' 101 (16,1,1,0)                                   : REM POSITION CUSOR
4208  PRINT       " DATA IMMPROPERLY STORED"                   : REM PRNT ERR MSG
4212              R8$ = "99"                                   : REM SET ERR CODE
4216              R7$ = "Subroutine 105"                       : REM SET ERR SOURCE
4220  GOSUB ' 000                                              : REM ERR PROCESSING
4224  GOTO        4272                                         : REM BRNCH TO EXIT
4228 REM **
4232              R0  = R0 + 252                               : REM INCR BUFR INDX
4236              R0$ = STR(G9$(64),3,2)                       : REM STOR NXT ADRS
4240  IF          R0  < 2016                THEN 4184          : REM BUFR NOT FULL
4244              E1  = 3                                      : REM SET FLAG
4248              R0  = 2016                                   : REM SET TO BUR MAX
4252  GOTO        4272                                         : REM BRANCH TO EXIT
4256 REM **
4260              R0 = FNR(VAL(R2$(1)))                        : REM CMPUTE BYTE STORED
4264  MAT COPY    R5$()<1,1> TO R1$()<R0,1>                    : REM CLR EOI BYTE
4268              E1 = 4                                       : REM SET FLAG
4272  RETURN                                                   : REM EXIT
```

```
4276 REM **
4280 REM **
4284 REM **  Subroutine  ********************************************** 1 2 9
4288 REM **  Abstract   >
4292 REM **    To Write storage buffer to disk updating
4296 REM **    disk index. The three operational modes are
4300 REM **       - 1  create new entry in index;
4304 REM **       - 2  add contents of buffer to exising data;
4308 REM **       - 3  erase data item from disk.
4312 REM **  Variables >
4316 REM **    I1     > item id          ! I2   > function code
4320 REM **    Q8$()  > transfer bufr    ! I3   > sub-sctr indx
4324 REM **    S0$    > nxt avaiabl sctr! S1$  > indx sctr adrs
4328 REM **    S2$    > sub-sctr indx    ! S3$  > adrs registr
4332 REM **
4336 REM ********************************************
4340  DEFFN ' 129 (I1,I2)                                  : REM ENTRY
4344 REM **
4348 REM **        !  SAVE  !  ADD   ! DELETE !
4352 REM **        !--------!--------!--------!
4356  ON I2 GOTO     4368  ,  4392  ,  4412               : REM BRANCH
4360 REM **        !--------!--------!--------!
4364 REM **
4368  GOSUB ' 130 (I1)                                    : REM INDEX LOOKUP
4372  IF          I4 = 0                  THEN 4380      : REM NON EXISTANT ITEM
4376  GOSUB ' 131                                        : REM ERASE DATA
4380  GOSUB ' 132                                        : REM SAVE DATA BUFR
4384  GOTO        4432                                   : REM BRANCH TO EXIT
4388 REM **
4392  GOSUB ' 130 (I1)                                    : REM INDEX LOOKUP
4396  GOSUB ' 133                                        : REM SKIP TO END
4400  GOSUB ' 132                                        : REM SAVE DATA BUFR
4404  GOTO        4432                                   : REM BRANCH TO EXIT
4408 REM **
4412  GOSUB ' 130 (I1)                                    : REM INDEX LOOKUP
4416  IF          I4 = 0                  THEN 4432      : REM NON-EXISTANT ITEM
4420  GOSUB ' 131                                        : REM ERASE DATA
4424  GOTO        4432                                   : REM BRANCH TO EXIT
4428 REM **
4432  RETURN                                             : REM EXIT
```

```
4436 REM **
4440 REM **
4444  REM **  Subroutine  ********************************************** 1 3 0
4448  REM **  Function >  To search disk index for item ID.
4452  REM **  Variables >
4456  REM **     I4     > itm srch flg ! I5      > indx sctr adrs
4460  REM **     R9     > scratch indx ! Q8$()   > index bufr
4464  REM **     I5$    > search ID    ! I1      > item ID
4468  REM **
4472  REM **********************************************
4476  DEFFN ' 130 (I4)                                          : REM ENTRY
4480              BIN(STR(S5$,1,1))=INT(I4/256)                 : REM CMPUT BINARY ID
4484              BIN(STR(S5$,2,1))=I4-INT(I4/256)*256:         REM CMPUT BINARY ID
4488  MAT SEARCH  Q6$(), =STR(S5$,1,2) TO R2$() STEP 2:         REM SRCH ID TABLE
4492              I4 = FNR(VAL(R2$(1)))                         : REM CMPUT TBL POSITION
4496  IF          I4 = 0                          THEN 4524     : REM NULL LOOKUP
4500              D9 = INT(I4/2) + 1                            : REM CMPUT TBL SUBSCRPT
4504              D0 = INT((D9-1)/64) + 1                       : REM CMPUT ROW SUBSCRPT
4508              D5 = D9 - (D0-1)*64                           : REM CMPUT COL SUBSCRPT
4512              D1 = INT(I4/128) + 1                          : REM CMPUT SCTR ADDRS
4516              D2 = I4 - ((D1-1)*128)                        : REM CMPUT SUBSCTR ADRS
4520  DATA LOAD   BA T #2, (D1,D9) Q8$()                        : REM LOAD INDX SCTR
4524  RETURN                                                    : REM EXIT
```

```
4528 REM **
4532 REM **
4536  REM **  Subroutine  ********************************************** 1 3 1
4540  REM **  Abstract >
4544  REM **    To erase existing data from disk. Also deleting
4548  REM **    entry from disk index and memory ID table.
4552  REM **  Variables >
4556  REM **    SO$    > nxt avable sctr! S1$    > indx sctr adrs
4560  REM **    S2$    > sub-sctr adrs  ! I3     > sub-sctr adrs
4564  REM **    Q8$()  > transfer bufr  ! I1     > item id
4568  REM **    S3$    > scratch regstr ! R2$    > scratch bufr
4572  REM **    R2$()  > search regstr  ! I4     > itm scrch flg
4576  REM **
4580  REM ***********************************************
4584  DEFFN ' 131                                          : REM ENTRY
4588              Q6$(D0,D5) = HEX(0000)                   : REM DELETE FROM ID TBL
4592              S5$ = Q8$(D2+1)                          : REM STOR BEG ADDRESS
4596              Q8$(D2) = HEX(0000)                      : REM DELETE FROM INDX
4600              Q8$(D2+1) = HEX(FFFF)                    : REM DELETE FROM INDX
4604  DATA SAVE   BA T #2, (D1,D9) Q8$()                   : REM UPDATE DISK
4608 REM **
4612  DATA LOAD   BA T #2, (S5$,S9$) Q9$()                 : REM LOAD DATA SECTOR
4616              S6$ = STR(Q9$(64),3,2)                   : REM STOR LINK KEY
4620  INIT(00)    Q9$()                                    : REM CLR DATA BUFFR
4624  DATA SAVE   BA T #2, (S5$,S9$) Q9$()                 : REM UPDATE DISK
4628  IF          S6$ = HEX(FFFF)           THEN 4644      : REM DELETE COMPLETED
4632              S5$ = S6$                                : REM UPDATE ADDRESS
4636  GOTO        4612                                     : REM LOAD NXT SCTR
4640 REM **
4644  RETURN                                               : REM EXIT
```

```
4648 REM **
4652 REM **
4656  REM **  Subroutine  ********************************************** 1 3 2
4660  REM **  Abstract >
4664  REM **    To copy data from storage buffer to disk. At end
4668  REM **    incrementing to next avaiable sector address. Also
4672  REM **    update index if transfer is the initial storage
4676  REM **    of the data buffer.
4680  REM **  Variables >
4684  REM **    S0$    > nxt avable sctr! S1$    > indx sctr adrs
4688  REM **    S2$    > sub-sctr adrs  ! I3     > sub-sctr adrs
4692  REM **    Q8$()  > transfer bufr  ! I1     > item id
4696  REM **    S3$    > scratch regstr ! R2$    > scratch bufr
4700  REM **    R2$()  > search regstr  ! I4     > itm scrch flg
4704  REM **    S4$    > currnt sctr adrs! I6    > out bufr indx
4708  REM **
4712  REM ***********************************************
4716  DEFFN ' 132                                          : REM ENTRY
4720  IF          I2 = 2                        THEN 4728  : REM CONTINUATION DATA
4724  GOSUB ' 134                                          : REM UPDATE INDEX
4728  MAT SEARCH  R1$(), = R5$(2) TO R2$()                 : REM SRCH FOR EOI CHAR
4732              I4 = FNR(VAL(R2$(1)))                    : REM CMPUT POSITION
4736              I5 = 1                                   : REM INIT START POS
4740              I4 = INT((I4-1)/252) + 1                 : REM CMPUT TRANSFR SIZ
4744  REM **
4748   FOR        I6 = 1 TO I4                             : REM BEGIN TRANSFR LOOP
4752   MAT COPY   R1$()<I5,252> TO Q9$()<1,252>            : REM LOAD TRANSFR BUFR
4756              STR(Q9$(64),3,2) = S2$                   : REM STOR LINK ADDRESS
4760  IF          I6 <> I4                      THEN 4768  : REM NOT END OF STOR
4764              Q9$(64) = HEX(FFFFFFFF)                  : REM END OF DATA LINK
4768   DATA SAVE  BA T #2, (S0$,S6$) Q9$()                 : REM TRANSFR TO DISK
4772   GOSUB ' 135 (S2$,HEX(0000))                         : REM FETCH NXT ADDRESS
4776              I5 = I5 + 252                            : REM INCR BUFR POSITION
4780   NEXT       I6                                       : REM CONT TRANSFR LOOP
4784  RETURN                                               : REM EXIT
```

```
4788 REM **
4792 REM **
4796 REM **  Subroutine  ********************************************** 1 3 3
4800 REM **  Abstract >
4804 REM **    To setup for a disk transfer so as to link data to
4808 REM **    the end of existing data on disk.
4812 REM **  Variables >
4816 REM **    S0$    > nxt avable sctr    ! S1$    > indx sctr adrs
4820 REM **    S2$    > sub-sctr adrs      ! I3     > sub-sctr adrs
4824 REM **    Q8$()  > transfer bufr      ! I1     > item id
4828 REM **    S3$    > scratch regstr     ! R2$    > scratch bufr
4832 REM **    R2$()  > search regstr      ! I4     > itm srch flg
4836 REM **
4840 REM ********************************************
4844  DEFFN ' 133                                          : REM ENTRY
4848          S6$ = Q8$(D2+1)                              : REM FETCH BEG SCTR ADR
4852  DATA LOAD  BA T #2, (S6$,S7$) Q9$()                 : REM LOAD DATA SCTR
4856          S7$ = STR(Q9$(64),3,2)                       : REM FETCH LINK ADRS
4860  IF       S7$ = HEX(FFFF)            THEN 4876       : REM END OF DATA ITEM
4864          S6$ = S7$                                    : REM SET LINK ADDRESS
4868  GOTO     4852                                         : REM CONT SEARCH
4872 REM **
4876          STR(Q9$(64),3,2) = S0$                       : REM UPDATE LINK ADDRES
4880  MAT SEARCH  Q9$(), R5$(2) TO R2$()                   : REM SRCH FOR EOI
4884          I4 = FNR(VAL(R2$(1)))                        : REM CMPUT EOI POSITION
4888  MAT COPY   R5$()<1,1> TO Q9$()<I4,1>                : REM CLR EOI CHARACTER
4892  DATA SAVE  BA T #2, (S6$,S7$) Q9$()                 : REM UPDATE DISK
4896  RETURN                                               : REM EXIT
```

```
4900 REM **
4904 REM **
4908 REM **  Subroutine  ********************************************** 1 3 4
4912 REM **  Abstract >
4916 REM **    To update disk index and id memory table with current
4920 REM **    item id and begining disk location.
4924 REM **  Variables >
4928 REM **    I4 > item lookup flag      ! I1 > item id
4932 REM **    Q6$() > memory id table    ! Q8$() > disk index buffer
4936 REM **    S0$ > begining location    ! D0 > id table index
4940 REM **    D2 > index sector indes    ! D1 > current index sctr adrs
4944 REM **
4948 REM ************************************************
4952  DEFFN ' 134                                          : REM ENTRY
4956  GOSUB ' 130 (0)                                      : REM INDEX LOOKUP
4960  IF          I4 <> 0                        THEN 4992 : REM SLOT FOUND
4964  GOSUB ' 101 (14,1,3,0)                               : REM POSITION CUSOR
4968  GOSUB ' 101 (15,10,0,0)                              : REM POSITION CUSOR
4972  PRINT       "System Error: Index overflow"           : REM ERROR MSG
4976  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
4980  GOSUB ' 101 (14,1,0,0)                               : REM POSITION CUSOR
4984  STOP        "** SYSTEM STOPPED **"                   : REM INTERUPT PROGRAM
4988 REM **
4992              BIN(STR(S1$,1,1)) = INT(I1/256)         : REM STOR BINARY ID
4996              BIN(STR(S1$,2,1))=I1-INT(I1/256)*256:   REM STOR BINARY ID
5000              Q6$(D0,D5) = S1$                        : REM UPDATE MEMORY TBL
5004              Q8$(D2) = S1$                           : REM UPDATE DSK IDX BUF
5008              Q8$(D2+1) = S0$                         : REM UPDATE DSK IDX BUF
5012  DATA SAVE   BA T #2, (D1,D9) Q8$()                  : REM UPDATE DISK
5016  RETURN                                               : REM EXIT
```

```
5020 REM **
5024 REM **
5028 REM **  Subroutine  ********************************************** 1 3 5
5032 REM **  Abstract >
5036 REM **    To reset address registers to the next avaiable
5040 REM **    sector address.
5044 REM **  Variables >
5048 REM **    S0$ > current sctr adrs    ! S7$ > next adrs (new)
5052 REM **    S6$ > storage register     ! S2$ > next adrs aftr S7$
5056 REM **    Q9$() > scratch buffer     !
5060 REM **
5064 REM ***********************************************
5068  DEFFN ' 135 (S0$,S7$)                                  : REM ENTRY
5072  DATA LOAD   BA T #2, (S0$,S6$) Q9$()                   : REM LOAD NEXT SCTR
5076  IF          Q9$(1) = HEX(00000000)     THEN 5116       : REM SCTR IS CLEAR
5080              S0$ = S6$                                  : REM INCR ADDRESS
5084  IF          S0$ <> HEX(03FF)           THEN 5072       : REM CHECK NEXT ADRS
5088  GOSUB ' 101 (14,1,3,0)                                 : REM POSITION CUSUR
5092  GOSUB ' 101 (15,10,0,0)                                : REM POSITION CUSOR
5096  PRINT       "System Error: Disk Limits Exceeded"       : REM ERROR MSG
5100  $GID TONE   /005 (R8$(4),R8$(9))                       : REM PAUSE TONE
5104  GOSUB ' 101 (14,1,0,0)                                 : REM POSITION CUSOR
5108  STOP        "** SYSTEM STOPPED **"                     : REM PROGRAM INTERUPT
5112 REM **
5116  IF          S7$ <> HEX(0000)           THEN 5136       : REM RESET COMPLETE
5120              S7$ = S0$                                  : REM STOR NEW ADDRS
5124  GOSUB ' 135 (S6$,S7$)                                  : REM CMPUT NEXT ADRS
5128              S2$ = S0$                                  : REM STOR NEW ADDRS
5132              S0$ = S7$                                  : REM RESET ADDRS
5136  RETURN                                                 : REM EXIT
```

```
5140 REM **
5144 REM **
5148 REM **  Subroutine  ********************************************** 1 3 6
5152 REM **  Abstract >
5156 REM **    This routine will locate the last character in the
5160 REM **    buffer and return the total length of the buffer
5164 REM **    in variable R0.
5168 REM **  Variables >
5172 REM **    R0 > bufr line indx       ! R1$() > text buffer
5176 REM **    R9$() > temp line bufr    ! R5$() > search characters
5180 REM **    R2$() > srch return bufr !
5184 REM **
5188 REM ***********************************************
5192  DEFFN ' 136 (R0)                                          : REM ENTRY
5196  MAT COPY   -R1$()<(R0-1)*63+1,63> TO R9$()<1,64>:: REM REVERSE COPY
5200  MAT SEARCH  R9$()<1,63>, <> R5$(1) TO R2$()      : REM SRCH FOR END
5204  IF          R2$(1) <> HEX(0000)        THEN 5216 : REM IF END FOUND
5208              R0 = R0 - 1                          : REM DCRMNT LINE INDX
5212  IF          R0 > 0                     THEN 5196 : REM BUFFR NOT EMPTY
5216              R0 = (32-R0)*63+(FNR(VAL(R2$(1)))-1):: REM DETERMIN BUFR SIZ
5220              R0 = 2016-R0                         : REM SCALE VALUE
5224  RETURN                                           : REM EXIT
```

AUTHOR System: Phase 2 - ATH3/I1 - # MAP

4076 - 4052
4100 - 4076
4184 - 4240
4232 - 4192
4260 - 4200
4272 - 4224 4252
4368 - 4356
4380 - 4372
4392 - 4356
4412 - 4356
4432 - 4384 4404 4416 4424
4524 - 4496
4612 - 4636
4644 - 4628
4728 - 4720
4768 - 4760
4852 - 4868
4876 - 4860
4992 - 4960
5072 - 5084
5116 - 5076
5136 - 5116
5196 - 5212
5216 - 5204

AUTHOR System: Phase 2 - ATH3/II - V MAP

```
D0     - 4504 4508 4588 5000
D1     - 4512 4516 4520 4604 5012
D2     - 4516 4592 4596 4600 4848 5004 5008
D5     - 4508 4588 5000
D9     - 4500 4504 4508 4520 4604 5012
E1     - 4244 4268
I1     - 4172 4340 4368 4392 4412 4992 4996
I2     - 4340 4356 4720
I4     - 4372 4416 4476 4480 4484 4492 4496 4500 4512 4516 4732 4740 4748 4760
         4884 4888 4960
I5     - 4736 4752 4776
I6     - 4748 4760 4780
Q6$(   - 4488 4588 5000
Q8$(   - 4520 4592 4596 4600 4604 4848 5004 5008 5012
Q9$(   - 4184 4188 4192 4236 4612 4616 4620 4624 4752 4756 4764 4768 4852 4856
         4876 4880 4888 4892 5072 5076
R0     - 4172 4180 4188 4232 4240 4248 4260 4264 5192 5196 5208 5212 5216 5220
R0$    - 4172 4184 4236
R0$(   - 4048 4060 4108
R1     - 4056 4100
R1$(   - 4060 4176 4188 4196 4264 4728 4752 5196
R2     - 4052 4056 4060 4064 4100
R2$(   - 4196 4200 4260 4488 4492 4728 4732 4880 4884 5200 5204 5216
R3     - 4060 4104
R3$    - 4184
R4     - 4076 4104
R5     - 4068 4104
R5$(   - 4196 4264 4728 4880 4888 5200
R7$    - 4216
R8$    - 4212
R8$(   - 4088 4092 4096 4976 5100
R9$(   - 5196 5200
S0$    - 4768 4876 5008 5068 5072 5080 5084 5120 5128 5132
S1$    - 4992 4996 5000 5004
S2$    - 4756 4772 5128
S5$    - 4480 4484 4488 4592 4612 4624 4632
S6$    - 4616 4628 4632 4768 4848 4852 4864 4892 5072 5080 5124
S7$    - 4852 4856 4860 4864 4892 5068 5116 5120 5124 5132
S9$    - 4612 4624
```

```
AUTHOR System: Phase 2 - ATH3/I1   - ' MAP

???? DEFFN' 0
       - 4220
???? DEFFN' 101
       - 4056 4080 4100 4204 4964 4968 4980 5088 5092 5104
???? DEFFN' 110
       - 4108
4044 DEFFN' 124
4172 DEFFN' 125
4340 DEFFN' 129
4476 DEFFN' 130
       - 4368 4392 4412 4956
4584 DEFFN' 131
       - 4376 4420
4716 DEFFN' 132
       - 4380 4400
4844 DEFFN' 133
       - 4396
4952 DEFFN' 134
       - 4724
5068 DEFFN' 135
       - 4772 5124
5192 DEFFN' 136
```

4.3 ATH2INIT

The purpose of the 'ATH2INIT' program is to initialize the program variables and the data base diskette. The program will initialize the diskette either for initial data entry or for maintenance of the data base.

The 'ATH2INIT' program is presented here in two parts.

4.3.1 ATH2INT1

This section of the 'ATH2INIT' program contains subroutines 2ØØ to 2Ø3.

The remainder of this section presents the 'ATH2INT1' program listing and reference maps.

AUTHOR System: Phase 2 - ATH2INT1 - INITL

```
6000  %## Program > AUTHOR/2  (ATH2INT1)   Version > 2.0  2200 T/VP  780320
6004  REM **  Abstract  ************************************************
6008  REM **                                                            **
6012  REM **    ATH2INIT will provide logic for initializing            **
6016  REM **    the AUTHOR Data Base Disk. When initializing            **
6020  REM **    is completed ATH2INIT will load logic routines          **
6024  REM **    for manipulating the Data Base.                         **
6028  REM **                                                            **
6032  REM ************************************************************
6036 REM **
6040 REM **
6044  GOSUB ' 200                                  : REM TBL INITIALIZATN
6048  GOSUB ' 201                                  : REM DISPLAY TITL PAG
6052  GOSUB ' 202                                  : REM DISPLAY COMMANDS
6056  GOSUB ' 203                                  : REM EXECUTE COMMANDS
6060  GOSUB ' 002                                  : REM LOAD LOGIC OVRLAY
```

```
6064 REM **
6068 REM **                                                    LABLED SUBROUTINES
6072 REM **
6076 REM **  Subroutine  ********************************************** 2 0 0
6080 REM **  Abstract >
6084 REM **    To initialize program Tables.
6088 REM **  Variables >
6092 REM **   R4$() > conversion tables
6096 REM **      (01)-(04) > <1,64>   lower case to upper case conv.
6100 REM **      (05)-(08) > <65,64>  upper case to lower case conv.
6104 REM **      (09)-(10) > <129,32> conv binary indx to fiche row char.
6108 REM **      (11) >      <161,16> control character conversion (input)
6112 REM **      (12) >      <177,16> buffer mode character conversions -
6116 REM **           >      <177,4>  HEX(00) to HEX(20)
6120 REM **           >      <181,4>  HEX(00) to HEX(00)
6124 REM **           >      <185,4>  HEX(30) to HEX(20)
6128 REM **           >      <189,4>  HEX(30) to HEX(00)
6132 REM **      (13)-(14) > <193,32> convert mask digits to normal codes
6136 REM **      (15)-(16) > <225,32> conv fiche row char. to binary indx
6140 REM **      (17)      > input character conversion (entry mode)
6144 REM **      (18)-(20) > input character conversion (edit mode)
6148 REM **   R5$() > single character search ids
6152 REM **      (01) > blank          (02) > HEX(FF)
6156 REM **      (03) > HEX(80)        (04) > HEX(5F)<underscor>
6160 REM **      (05) > HEX(3A)<coln>  (06) > HEX(3C) < < >
6164 REM **      (07) > HEX(3E)< > >   (08) > HEX(2E) < . >
6168 REM **      (09) > HEX(20)<blnk>  (10) > HEX(23) < # >
6172 REM **   R6$() > cusor control <vert.>
6176 REM **   R7$() > cusor control <horz.>
6180 REM **   K5$() > cusor contorl <erase>
6184 REM **   R8$() > micro command buffer & output sequences
6188 REM **      (01) > cusor blink w/  overprint char.
6192 REM **      (02) > output of buffer with CRLF
6196 REM **      (03) >
6200 REM **      (04) > warning tone w/ 2 sec wait
6204 REM **      (05) >
6208 REM **      (06) > output 16 line buffer to crt
6212 REM **      (07) > single tone
6216 REM **      (08) > set register 5 with CR or LF
6220 REM **      (09) > command register buffer
6224 REM **
6228 REM ********************************************
6232  DEFFN ' 200                                          : REM ENTRY
```

```
6232  DEFFN ' 200                                                    : REM ENTRY
6236  R4$(01)=  HEX(41614262436344644565466647674868)
6240  R4$(02)=  HEX(49694A6A4B6B4C6C4D6D4E6E4F6F5070)
6244  R4$(03)=  HEX(51715272537354745575567657775878)
6248  R4$(04)=  HEX(59795A7A20202020202020202020202020)
6252  R4$(05)=  HEX(61416242634364446545664667476848)
6256  R4$(06)=  HEX(69496A4A6B4B6C4C6D4D6E4E6F4F7050)
6260  R4$(07)=  HEX(71517252735374547555765677577858)
6264  R4$(08)=  HEX(79597A5A20202020202020202020202020)
6268  R4$(09)=  HEX(204142434445464748494A4B4C4D4E4F)
6272  R4$(10)=  HEX(20202020202020202020202020202020)
6276  R4$(11)=  HEX(010802A7030D03E620202020202020)
6280  R4$(12)=  HEX(200020200000202020302020003020)
6284  R4$(13)=  HEX(30103111321233133414351536163717)
6288  R4$(14)=  HEX(38183919200002020202020202020202020)
6292  R4$(15)=  HEX(000102030405060708090A0B0C0D0E0F)
6296  R4$(16)=  HEX(20202020202020202020202020202020)
6300  R4$(17)=  HEX(010802A703E504E6050D20202020202020)
6304  R4$(18)=  HEX(01A001080BCB025E0CCA03280DC70429)
6308  R4$(19)=  HEX(0EC3052A05C2062D06C4072B07C6090D)
6312  R4$(20)=  HEX(0AE5082F08CC022020202020202020)
6316  R5$(01)=  HEX(20)          :       R5$(02)= HEX(FF)
6320  R5$(03)=  HEX(80)          :       R5$(04)= HEX(5F)
6324  R5$(05)=  HEX(3A)          :       R5$(06)= HEX(3C)
6328  R5$(07)=  HEX(3E)          :       R5$(08)= HEX(2E)
6332  R5$(09)=  HEX(20)          :       R5$(10)= HEX(23)
6336  INIT(0A)  R6$()            :       R6$(01)= HEX(01)
6340  INIT(09)  R7$()            :       R7$(01)= HEX(0D)
6344  INIT(20)  K5$()            :       K5     = 1
6348  R8$(01)=  HEX(400642604008121142504008113614541608114 6)
6352  R8$(02)=  HEX(4006A0004021400D400A2020)
6356  R8$(03)=  HEX(FFFF2020)
6360  R8$(04)=  HEX(40071221400712214007 2020)
6364  R8$(05)=  HEX(FFFF2020)
6368  R8$(06)=  HEX(A000400D1427160840 0A2020)
6372  R8$(07)=  HEX(400740072020)
6376  R8$(08)=  HEX(050D14671708050A2020)
6380  R8$(09)=  HEX(05163F2055667700 0000)
6384  REM **
6388  Q1$(1)=   "TITLE"          : Q1$(6)=   "TRAINING"    : Q1$(12)=   "EXAMPLE"
6392  Q1$(2)=   "SUBTITLE"       : Q1$(7)=   "IMPORTANCE"  : Q1$(13)=   "SYMBOL"
6396  Q1$(3)=   "AUTHORS"        : Q1$(8)=   "USAGE"       : Q1$(14)=   "HELP"
6400  Q1$(4)=   "DATE"           : Q1$(9)=   "TYPE"        : Q1$(15)=   "LIST"
6404  Q1$(5)=   "ORGANIZTION"    : Q1$(10)=  "NAME"        : Q1$(16)=   "ESCAPE"
6408                               Q1$(11)=  "SOURCE"      : Q1$(17)=   "??????"
6412  REM **
6416  Q7$(1)=   "INITIAL"        : Q7$(4)=   "ASSIGN"
6420  Q7$(2)=   "MAINTENANCE"    : Q7$(5)=   "HELP"
6424  Q7$(3)=   "BACKUP"
6428  REM **
6432  Q5$(1)=   "SETNUMBER"      : Q5$(5)=   "SYMBOL"      : Q5$(09)=   "NEXT"
6436  Q5$(2)=   "SBLNUMBER"      : Q5$(6)=   "DEFINITION"  : Q5$(10)=   "DELETE"
6440  Q5$(3)=   "SETNAME"        : Q5$(7)=   "MNEMONIC"    : Q5$(11)=   "EXIT"
6444  Q5$(4)=   "REFERENC #"     : Q5$(8)=   "SIMILARSBL"  : Q5$(12)=   "LIST"
6448  REM **
```

```
6452  H8$( 1)=  "AUTHOR: An Automatic Authoring System     2.0"
6456  H8$( 2)=  "Volume ID:  Phase 2 Data Base"
6460  H8$( 3)=  "page:"
6464  H8$( 4)=  "1.1  Title:"
6468  H8$( 5)=  "1.2  Subtitle:"
6472  H8$( 6)=  "1.3  Author(s):"
6476  H8$( 7)=  "1.4  Date:"
6480  H8$( 8)=  "1.5  Organization:"
6484  H8$( 9)=  "2.0  Learning Objective:"
6488  H8$(10)=  "3.1  Importance of Symbols:"
6492  H8$(11)=  "3.2  How the Skill Will be Used:"
6496  H8$(12)=  "4.1  Name of the User:"
6500  H8$(13)=  "4.2  Type of Symbol:"
6504  H8$(14)=  "4.3  Official Source of Reference:"
6508  H8$(15)=  "5.1  Example of Symbols:"
6512  H8$(16)=  "6.1  Description of Symbols:"
6516  H8$(17)=  "          Set Name:"
6520  H8$(18)=  "          Set #:            Symbol #:"
6524  H8$(19)=  "          Symbol Code:      Symbol:"
6528  H8$(20)=  "          Definition:"
6532  H8$(21)=  "          Memory Aid:"
6536  H8$(22)=  "          Similar Symbols:"
6540  H8$(23)=  "          Symbol Code:"
6544 REM **
6548  N1$   =   "TITLE"       : N2$   =   "SETNUMBER"
6552 REM **
6556  RETURN                                              : REM EXIT
```

```
6560 REM **
6564 REM **
6568  REM **  Subroutine  ********************************************** 2 0 1
6572  REM **  Abstract   >
6576  REM **    To display Title page on console device.
6580  REM **  Variables  >
6584  REM **    R0$() > input char bufr! Q2$() > cmnd indx bufr
6588  REM **    R8$() > micro cmnd bufr!
6592  REM **
6596  REM ***********************************************
6600  DEFFN ' 201                                          : REM ENTRY
6604  SELECT  PRINT 005                                    : REM SEL OUTPUT DISPLY
6608  PRINT   HEX(03)                                      : REM CLR CRT SCREEN
6612  PRINT   "Program  > AUTHOR/2"
6616  PRINT   "Version  > 2.0"
6620  PRINT   "Abstract >"
6624  PRINT   "    This Program will enable you to create and"
6628  PRINT   "    maintain the AUTHOR System Data Base."
6632  PRINT   "    "
6636  PRINT   "    For detailed program instructions see the "
6640  PRINT   "    AUTHOR System Data Entering User's Guide."
6644  SELECT  PRINT 405
6648  GOSUB ' 101 (15,1,0,0)                               : REM POSITION CUSOR
6652              STR(R8$(9),5,1) = HEX(20)                : REM SET CUSOR OPTION
6656  PRINT   "Touch 'RETURN' to Continue With Program > ";
6660  KEYIN       R0$(1), 6672, 6660
6664  $GIO CUSOR  /005(R8$(1),R8$(9))                      : REM RESPONCE WAIT
6668  GOTO        6660                                     : REM CONT INPT WAIT
6672  RETURN                                               : REM EXIT
```

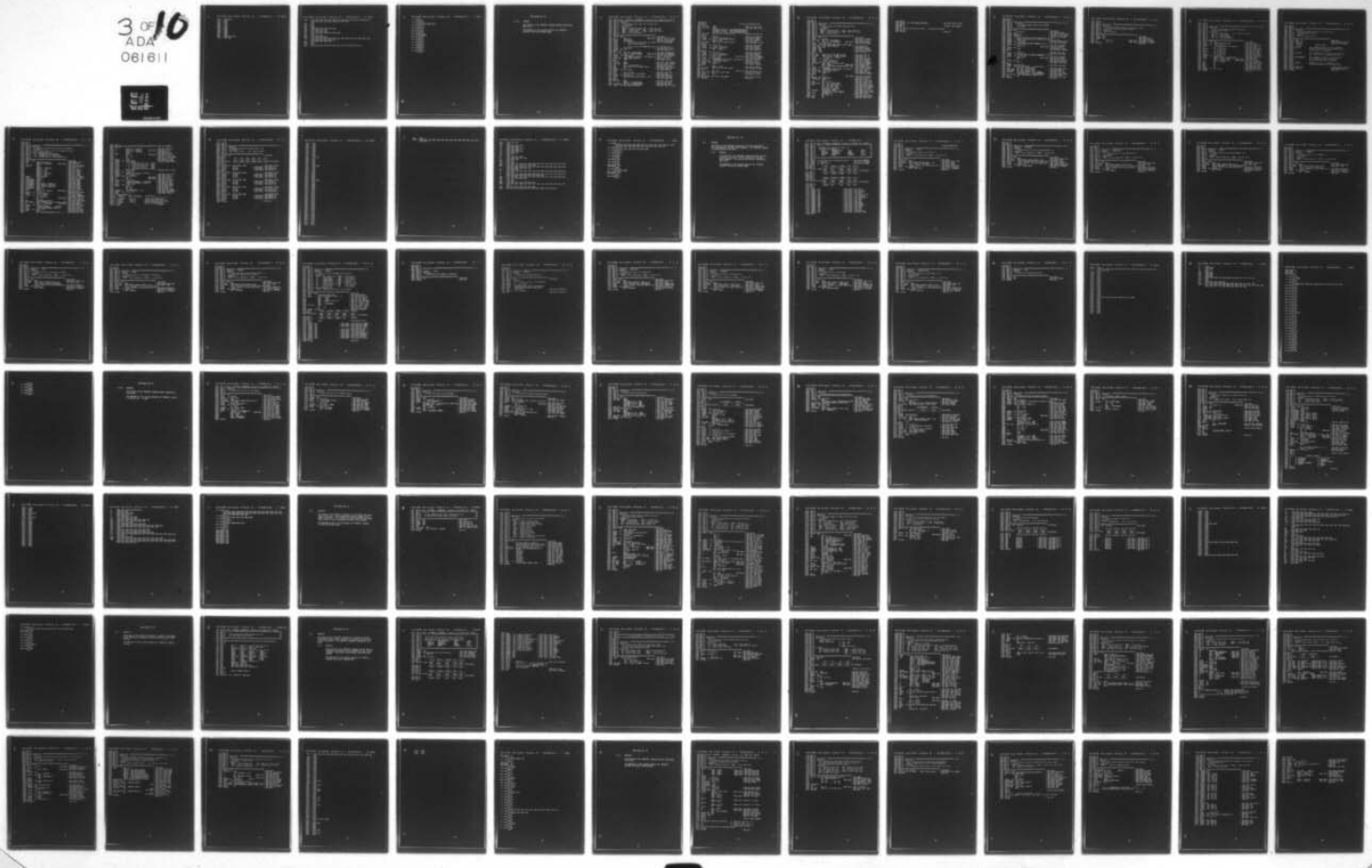
```
6676 REM **
6680 REM **
6684  REM ** Subroutine  ********************************************** 2 0 2
6688  REM **  Abstract   >
6692  REM **    To display the command summary and accept
6696  REM **    input command.
6700  REM **  Variables  >
6704  REM **    R2$    > input rtrn val! R8$() > micro cmnd bufr
6708  REM **    R2     > scratch index ! R0$() > char entry bufr
6712  REM **    R0     > command index ! Q2$() > cmnd index bufr
6716  REM **
6720  REM ***********************************************
6724  DEFFN ' 202                                          : REM ENTRY
6728  SELECT  PRINT 005                                    : REM SEL OUTPUT DISPLY
6732  PRINT   HEX(03)                                      : REM CLR CRT SCREEN
6736  PRINT   "Command Summary:"
6740  PRINT   "  ASSIGN       - Assign Disk Devices"
6744  PRINT   "  INITIAL      - Initial Entry of Data Base"
6748  PRINT   "  MAINTENANCE  - Maintenance of Data Base"
6752  PRINT   "  BACKUP       - Backup the Data Base"
6756  PRINT   "  HELP         - Sequence of Commands"
6760  SELECT  PRINT 405                                    : REM SEL OUTPUT DISPLY
6764  GOSUB ' 101 (15,1,0,0)                               : REM POSITION CUSOR
6768  PRINT       "Enter Command > "                       : REM PRINT PROMT
6772  GOSUB ' 104 (15,"A","HELP")                          : REM ENTER COMMAND
6776              R0 = LEN(R2$)                            : REM CMPUT STRNG LEN
6780  IF            R0 <= 2                       THEN 6788: REM WITHEN LIMITS
6784              R0 =  2                                  : REM SET TO MAX LEN
6788  AND           (STR(R2$,1,R0),DF)                     : REM MASK OUT LWR CASE
6792  MAT SEARCH  Q7$(), =STR(R2$,1,R0) TO R2$() STEP 2:   REM SRCH KEYWORD TBL
6796  IF            R2$(2) <> HEX(0000)           THEN 6804: REM INVALID COMMAND
6800  IF            R2$(1) <> HEX(0000)           THEN 6828: REM COMMAND FOUND
6804  GOSUB ' 101 (16,1,0,0)                               : REM INVALID COMMAND
6808  PRINT       " INVALID COMMAND    RE-ENTER"           : REM PRINT ERR MSG
6812  $GID TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
6816  GOSUB ' 101 (16,1,1,0)                               : REM ERASE MSG
6820  GOTO        6764                                     : REM BRNCH TO CMD INPT
6824 REM **
6828              R0 = INT(FNR(VAL(R2$(1)))/2) + 1         : REM CMPUT CMND INDEX
6832  RETURN                                               : REM EXIT
```

```
6836 REM **
6840 REM **
6844  REM **  Subroutine  ********************************************** 2 0 3
6848  REM **  Abstract >
6852  REM **    To Process the inputted Command.
6856  REM **  Variables >
6860  REM **    RO  > command index  !
6864  REM **
6868  REM ***********************************************
6872  DEFFN ' 203                                          : REM ENTRY
6876 REM **
6880 REM **        ! INITIAL ! MTNNCE ! BACKUP ! ASSIGN ! HELP !
6884 REM **        !---------!--------!--------!--------!------!
6888  ON RO GOTO     6900   ,  6904  ,  6908  ,  6912  , 6916 : REM BRANCH
6892 REM **        !---------!--------!--------!--------!------!
6896 REM **
6900  GOSUB ' 204                          : GOTO 6924 : REM INITIALIZE DISK
6904  GOSUB ' 205                          : GOTO 6924 : REM DATA MAINTENANCE
6908  GOSUB ' 206                          : GOTO 6920 : REM DATA BASE BACKUP
6912  GOSUB ' 207                          : GOTO 6920 : REM DISK ASSIGNMENT
6916  GOSUB ' 209                          : GOTO 6920 : REM DISPLAY HELP FILE
6920  GOSUB ' 202                          : GOTO 6888 : REM ENTER NEW CMND
6924  RETURN                                           : REM EXIT
```

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U).
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

3 OF 10
ADA
061611



AUTHOR System: Phase 2 - ATH2INT1 - # MAP

```
6660  - 6660 6668
6672  - 6660
6764  - 6820
6788  - 6780
6804  - 6796
6828  - 6800
6888  - 6920
6900  - 6888
6904  - 6888
6908  - 6888
6912  - 6888
6916  - 6888
6920  - 6908 6912 6916
6924  - 6900 6904
```

AUTHOR System: Phase 2 - ATH21NT1 - V MAP

```
H8$(   - 6452 6456 6460 6464 6468 6472 6476 6480 6484 6488 6492 6496 6500 6504
         6508 6512 6516 6520 6524 6528 6532 6536 6540
K5     - 6344
K5$(   - 6344
N1$    - 6548
N2$    - 6548
Q1$(   - 6388 6392 6396 6400 6404 6408
Q5$(   - 6432 6436 6440 6444
Q7$(   - 6416 6420 6424 6792
R0     - 6776 6780 6784 6788 6792 6828 6888
R0$(   - 6660
R2$    - 6776 6788 6792
R2$(   - 6792 6796 6800 6828
R4$(   - 6236 6240 6244 6248 6252 6256 6260 6264 6268 6272 6276 6280 6284 6288
         6292 6296 6300 6304 6308 6312
R5$(   - 6316 6320 6324 6328 6332
R6$(   - 6336
R7$(   - 6340
R8$(   - 6348 6352 6356 6360 6364 6368 6372 6376 6380 6652 6664 6812
```

AUTHOR System: Phase 2 - ATH21NT1 - ' MAP

```
???? DEFFN' 2
       - 6060
???? DEFFN' 101
       - 6648 6764 6804 6816
???? DEFFN' 104
       - 6772
6232 DEFFN' 200
       - 6044
6600 DEFFN' 201
       - 6048
6724 DEFFN' 202
       - 6052 6920
6872 DEFFN' 203
       - 6056
???? DEFFN' 204
       - 6900
???? DEFFN' 205
       - 6904
???? DEFFN' 206
       - 6908
???? DEFFN' 207
       - 6912
???? DEFFN' 209
       - 6916
```

4.3.2 ATH2INT2

This section of the 'ATH2INIT' program contains subroutines 204 to 211.

The remainder of this section presents the 'ATH2INT2' program listing and reference maps.

```
7000  %## Program > AUTHOR/2   (ATH2INT2)   Version > 2.0  22001/VP 15-Dec-77
7004  REM **  Subroutine  ********************************************** 2 0 4
7008  REM **  Abstract >
7012  REM **    To initialize disk index for initial entry
7016  REM **    of Data Base.
7020  REM **  Variables >
7024  REM **    R0$() > input char bufr! R1   > indx sctr adrs
7028  REM **    Q8$() > transfer bufr   ! R2   > adrs register
7032  REM **    R9$() > scratch bufr    ! R2$  > input str value
7036  REM **    R8$() > micro cmnd bufr!
7040  REM **
7044  REM **********************************************
7048  DEFFN ' 204                                          : REM ENTRY
7052  IF          R0 = 3                        THEN 7200  : REM CHG SYS SCTR ONLY
7056  SELECT      PRINT 005                                : REM SEL OUTPUT DISPLY
7060  PRINT       HEX(030A0A)                              : REM CLR CRT SCREEN
7064  PRINT       "    The Phase II Data Base Disk MUST first be FORMATTED"
7068  PRINT       "    Before Continueing."
7072  PRINT
7076  PRINT       "    Once the Disk is FORMATTED Touch the 'RETURN' Key."
7080  SELECT      PRINT 405                                : REM SEL OUTPUT DISPLY
7084  GOSUB ' 208 (15)                                     : REM RESPONCE WAIT
7088 REM **
7092  DATA LOAD   BAT #2, (0,R2) Q9$()                     : REM LOAD SYSTEM SCTR
7096  IF          Q9$(1) = HEX(00000000)        THEN 7120  : REM DSK IS FORMATED
7100  GOSUB ' 101 (15,1,1,30)                              : REM DSK NOT FORMATTED
7104  PRINT       "DISK DOES NOT VERIFY... RE-FORMAT..":   REM PRINT ERR MSG
7108  $GID TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
7112  GOTO        7084                                     : REM BRNCH TO WAIT
7116 REM **
7120  INIT(FF)    Q8$()                                    : REM INIT INDEX BUFFER
7124  INIT(00)    R2$()                                    : REM INIT SCRTCH BUFR
7128  FOR         R1 = 1 TO 256 STEP 4                     : REM BEG BUFR INIT
7132  MAT         COPY R2$()<1,2> TO Q8$()<R1,2>           : REM SETUP INDX BUFFR
7136  NEXT        R1                                       : REM CONT BUFR INIT
7140  GOSUB ' 101 (15,1,1,0)                               : REM ERASE SCREEN
7144 REM **
7148  FOR         R1 = 1 TO 10                             : REM BEG DISK LOAD
7152  DATA        SAVE BAT #2, (R1,R2)Q8$()                : REM LOAD DISK
7156  GOSUB ' 101 (15,1,0,0)                               : REM POSITION CUSOR
7160  PRINT       "INITIALIZING DISK SECTOR > ";R1         : REM PRINT MESSAGE
7164  NEXT        R1                                       : REM CONT DISK LOAD
7168 REM **
7172  INIT(EE)    Q8$()                                    : REM INIT INDX BUFR
7176              Q8$(1) = HEX(2003FF00)                   : REM SET EOF INFO
7180  DATA SAVE   BAT #2, (1023,R2)Q8$()                   : REM SAVE EOF SCTR
7184  GOSUB ' 135 (HEX(0008),HEX(0000))                    : REM SRCH FOR NXT SCTR
```

```
7188 REM **
7192 REM **                                          INITIALIZE SYSTEM SECTOR
7196 REM **
7200  INIT(00)     Q8$()                                    : REM INIT INDX BUFR
7204  INIT(00)     Q6$()                                    : REM INIT MEM ID TBL
7208               STR(R9$(1),01,08) = HEX(000503FF04000000)
7212               STR(R9$(1),09,08) = HEX(EEEEEEEEEEEEEEEE)
7216               STR(R9$(1),17,16) = HEX(10000000803FF0000)
7220  MAT COPY     R9$()<1,32> TO Q8$()<1,32>               : REM SET SYSTEM SCTR
7224 REM **
7228  PRINT        HEX(03)                                  : REM CLR CRT SCREEN
7232  PRINT        "Enter Disk Volume ID > "                : REM PRINT PROMPT
7236  GOSUB ' 104 (16,"A","AUTHOR Data Base")               : REM INPUT RESPONCE
7240               R9$(1)= R2$                              : REM STOR INPUT
7244  MAT COPY     R9$()<1,16> TO Q8$()<65,16>              : REM TRANSFER TO BUFR
7248 REM **
7252  GOSUB ' 101 (4,1,1,0)                                 : REM POSITION CUSOR
7256  PRINT        "Enter  Password > "                     : REM PRINT PROMPT
7260  GOSUB ' 104 (8,"E","password")                        : REM INPUT RESPONCE
7264               R3$   = R2$                              : REM STOR PASWRD 1
7268 REM **
7272  GOSUB ' 101 (4,1,1,0)                                 : REM POSITION CUSOR
7276  PRINT        "Verify Password > "                     : REM PRINT PROMPT
7280  GOSUB ' 104 (8,"E","password")                        : REM INPUT RESPONCE
7284  IF           R3$  = R2$                  THEN 7312    : REM INVALID PASSWRDS
7288  GOSUB ' 101 (4,28,0,0)                                : REM POSITION CUSOR
7292  PRINT        "PASSWORDS DO NOT MATCH  TRY AGAIN!"     : REM PRINT ERR MSG
7296  $GIO TONE    /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
7300  GOSUB ' 101 (4,28,1,0)                                : REM ERASE ERR MSG
7304  GOTO         7252                                     : REM REENTR PASWRDS
7308 REM **
7312               R9$(1) = R2$                             : REM STOR PASSWORD
7316  MAT COPY     R9$()<1,8> TO Q8$()<25,8>                : REM TRANFER TO BUFR
7320 REM **
7324  IF           R0 <> 3                     THEN 7340    : REM NOT BACKUP CALL
7328  DATA SAVE    BA T #3, (0,R2) Q8$()                    : REM SAVE SYS SCTR
7332  GOTO         7344                                     : REM BRANCH TO EXIT
7336 REM **
7340  DATA SAVE    BA T #2, (0,R2)Q8$()                     : REM SAVE SYS SCTR
7344  RETURN                                                : REM EXIT
```

```
7348 REM **
7352 REM **
7356  REM **  Subroutine  ********************************************** 2 0 5
7360  REM **  Abstract >
7364  REM **    To verify that the proper Data Base Disk
7368  REM **    is Mounted.
7372  REM **  Variables >
7376  REM **    Q8$() > transfer bufr    ! R2  > adrs register
7380  REM **    R8$() > micro cmnd bufr  ! R2$ > input str value
7384  REM **    R9$() > scratch bufr     !
7388  REM **
7392  REM **********************************************
7396  DEFFN ' 205                                                : REM ENTRY
7400  PRINT       HEX(03)                                        : REM CLR CRT SCREEN
7404  DATA LOAD   BA T #2, (0,R1)Q9$()                           : REM LOAD SYS SCTR
7408  IF          Q9$(3) = HEX(EEEEEEEE)      THEN 7444          : REM PROPER FORMAT
7412  GOSUB ' 101 (15,1,0,0)                                     : REM POSITION CUSOR
7416  PRINT   "DISK NOT AN AUTHOR PHASE II DATA BASE..."
7420  $GIO TONE   /005 (R8$(4),R8$(9))                           : REM PAUSE TONE
7424  GOSUB ' 208 (16)                                           : REM RESPONCE WAIT
7428  GOSUB ' 202                                                : REM INPUT COMMAND
7432  GOSUB ' 203                                                : REM EXECUTE COMMAND
7436  GOTO    7396                                               : REM TRY AGAIN
7440 REM **
7444  MAT COPY    Q9$()<65,16> TO R9$()<1,64>                    : REM FETCH VOL ID
7448  MAT COPY    Q9$()<65,16> TO H8$()<58,32>                   : REM STORE VOL ID
7452  PRINT       " Volume ID > ";R9$(1)                         : REM PRINT VOL ID
7456  GOSUB ' 101 (4,1,0,0)                                      : REM POSITION CUSOR
7460  PRINT       " Enter Password > "                           : REM PRINT PROMPT
7464  GOSUB ' 104 (8,"E","password")                             : REM INPUT RESPONCE
7468  MAT COPY    Q9$()<25,8> TO R9$()<1,64>                     : REM FETCH PASSWORD
7472  IF          R2$ = R9$(1)                    THEN 7504      : REM PASWRDS MATCH
7476  IF          R2$ = HEX(3C2074616567203E)THEN 7504           : REM PASWRDS MATCH
7480  GOSUB ' 101 (4,28,0,0)                                     : REM INVALID PASWORD
7484  PRINT       "PASSWORDS DO NOT MATCH  TRY AGAIN!"           : REM PRINT ERR MSG
7488  $GIO TONE   /005 (R8$(4),R8$(9))                           : REM PAUSE TONE
7492  GOSUB ' 101 (4,28,1,0)                                     : REM ERASE ERR MSG
7496  GOTO        7456                                           : REM REENTER PASWORD
7500 REM **
7504  IF          R0 = 3                          THEN 7596      : REM FNCT IS BACKUP
7508  GOSUB ' 101 (10,1,0,0)                                     : REM POSITION CUSOR
7512  PRINTUSING  7588, "> ";                                    : REM PRINT INDEX
7516              Q5 = 1                                         : REM SET ID TBL INDX
7520  FOR         R1 = 1 TO 10                                   : REM BEGIN INDX LOOP
7524  DATA LOAD   BA T #2, (R1,R2) Q8$()                         : REM LOAD INDX SCTR
7528  FOR         R2 = 1 TO 128 STEP 2                           : REM BEG SUBSCTR LOOP
7532              Q6 = INT((Q5-1)/64) + 1                        : REM CMPUT ROW SUBSRPT
7536              Q7 = Q5 - (Q6-1)*64                            : REM CMPUT COL SUBSCPT
7540              Q6$(Q6,Q7) = Q8$(R2)                           : REM LOAD ID TABLE
7544  CONVERT     Q5 TO R2$, (###)                               : REM CONV IDX TO DSPLY
7548              STR(R2$,4,3) = HEX(080808)                     : REM INSRT REPOS CHARS
7552  PRINT       STR(R2$,1,6)                                   : REM PRINT INDX
7556              Q5 = Q5 + 1                                    : REM INCR TBL INDX
7560  NEXT        R2                                             : REM CONT SUBIDX LOOP
7564  NEXT        R1                                             : REM CONT SCTR LOOP
```

```
7568 REM **
7572  GOSUB ' 135 (HEX(000B),HEX(0000))                    : REM TEST FOR CLR SCTR
7576 REM **
7580 REM **                                                   OUTPUT IMAGE FORMAT
7584 REM **
7588  %  Loading Internal ID Table...  Processing Item ##
7592 REM **
7596  RETURN                                                 : REM EXIT
```

```
7600 REM **
7604 REM **
7608  REM **  Subroutine  ********************************************** 2 0 6
7612  REM **    Abstract >
7616  REM **      To create a Backup Copy of the AUTHOR
7620  REM **      Data Base.
7624  REM **
7628  REM ***********************************************
7632  DEFFN ' 206                                          : REM ENTRY
7636  DIM         B1$(4)64                                 : REM DEFINE BUFRS
7640  GOSUB ' 205                                          : REM VERIFY AUTHOR DSK
7644  SELECT  PRINT 005                                    : REM SEL OUTPUT DISPLY
7648  PRINT   HEX(03)                                      : REM CLR CRT SCREEN
7652  PRINT   "  Verify that Phase II Data Base Disk and the"
7656  PRINT   "  Phase II Backup Disk are mounted in proper drives"
7660  SELECT      PRINT 405                                : REM SEL CONSOLE MODE
7664  GOSUB ' 101 (15,1,0,0)                               : REM POSITION CUSOR
7668  PRINT       "ARE DISKS PROPERLY MOUNTED (Y/N)? ";
7672  GOSUB ' 104 (1,"A","Y")                              : REM INPUT RESPONCE
7676  IF          R2$ = "Y"                    THEN 7692   : REM BEGIN BAKUP
7680  GOSUB ' 207                                          : REM REASSIGN DISKS
7684  GOTO        7632                                     : REM BRANCH TO BEGING
7688 REM **
7692              R0 = 0                                   : REM SET COPY INDX
7696  GOSUB ' 101 (15,1,1,0)                               : REM POSITION CUSOR
7700  PRINT       "DO YOU WISH TO CHANGE PASSWORDS (Y/N)? ";
7704  GOSUB ' 104 (1,"A","Y")                              : REM INPUT RESPONCE
7708  IF          R2$ <> "Y"                   THEN 7728   : REM DO NOT CHANGE
7712              R0 = 3                                   : REM SET FUNCTION FLG
7716  GOSUB ' 204                                          : REM CHANGE PASSWORD
7720              R0 = 1                                   : REM RESET COPY INDX
7724 REM **
7728  GOSUB ' 101 (8,1,0,0)                                : REM POSITION CUSOR
7732  PRINT   "COPYING DATA BASE . . . Disk Sector > "
7736  FOR         R2 = R0 TO 1023 STEP 1                   : REM BEGIN COPY LOOP
7740  DATA LOAD   BA T #2, (R2,R3) B1$()                   : REM LOAD BUFR #1
7744  DATA SAVE   BA T$#3, (R2,R3) B1$()                   : REM SAVE BUFR #1
7748  CONVERT     R2 TO STR(R9$(1),1,5), (#####)           : REM CONV SCTR INDX
7752              STR(R9$(1),6,5) = HEX(0808080808)        : REM STOR REPOS CHARS
7756  $G10 PRINT  /005 (A000,R8$(9)) R9$()<1,10>           : REM PRINT INDX
7760  NEXT        R2                                       : REM CONT COPY LOOP
7764  RETURN                                               : REM EXIT
```

```
7768  REM **
7772  REM **
7776  REM **  Subroutine  ********************************************** 2 0 7
7780  REM **  Abstract >
7784  REM **    To assign an input disk device to device table
7788  REM **    and verify disk has proper contents.
7792  REM **  Variables >
7796  REM **    R0 > eof flag          !
7800  REM **
7804  REM ***********************************************
7808  DEFFN ' 207                                          : REM ENTRY
7812  GOSUB ' 210                                          : REM INPUT DEV #
7816  GOSUB ' 211                                          : REM ASSIGN DEVICE
7820  IF          R0 <> 4                    THEN 7812     : REM MORE TO ASSIGN
7824  IF          R1 <> 6                    THEN 7812     : REM MORE TO ASSIGN
7828  RETURN                                               : REM EXIT
```

```
7832  REM **
7836  REM **
7840  REM **  Subroutine  ********************************************** 2 0 8
7844  REM **  Abstract >
7848  REM **    To wait for operator responce displaying
7852  REM **    elapsed time of wait.
7856  REM **  Variables >
7860  REM **    JO$, JO > time hours      ! K1 > row for display
7864  REM **    J1$, J1 > time mintues    !
7868  REM **    J2$, J2 > time seconds    !
7872  REM **    J3$, J3 > time adjustment!
7876  REM **
7880  REM ************************************************
7884  DEFFN ' 208  (K1)                                          : REM ENTRY
7888  DIM          JO$2,J1$2,J2$2,J3$3                           : REM INITIALIZE
7892  GOSUB ' 101  (K1,1,1,0)                                    : REM POSITION CUSOR
7896  PRINT        " Touch 'RETURN' to Continue"                 : REM PRNT INST.
7900  FOR          JO = 0 TO 2                                   : REM BEG HOURS LOOP
7904  CONVERT      JO TO JO$, (##)                               : REM CONV TO DISPLAY
7908  FOR          J1 = 0 TO 59                                  : REM BEG MINS LOOP
7912  CONVERT      J1 TO J1$, (##)                               : REM CONV TO DISPLAY
7916  FOR          J2 = 0 TO 59                                  : REM BEG SEC LOOP
7920  CONVERT      J2 TO J2$, (##)                               : REM CONV TO DISPLAY
7924  GOSUB ' 101  (K1,50,0,0)                                   : REM POSITION CUSOR
7928  PRINT        JO$;":";J1$;":";J2$;HEX(06)                   : REM PRNT CLOCK
7932  KEYIN        RO$(1), 7936, 7936      : GOTO 7940           : REM INPUT EOW
7936               JO=2 :  J1=59: J2=59    : GOTO 7948           : REM END OF LOOPS
7940  FOR          J3 = 0 TO 500                                 : REM BEG TIMMNG LOOP
7944  NEXT         J3                                            : REM CONT TIM CLOCK
7948  NEXT         J2                                            : REM CONT SEC CLOCK
7952  NEXT         J1                                            : REM CONT MIN CLOCK
7956  NEXT         JO                                            : REM CONT HRS CLOCK
7960  GOSUB ' 101  (K1,1,1,0)                                    : REM POSITION CUSOR
7964  RETURN                                                     : REM EXIT
```

```
7968  REM **
7972  REM **
7976  REM **  Subroutine  ********************************************** 2 0 9
7980  REM **  Abstract >
7984  REM **    Provide the basic sequence for initializing
7988  REM **    the Data Entry Process.
7992  REM **  Variables >
7996  REM **    RO$() > input char buffr !
8000  REM **
8004  REM ********************************************
8008  DEFFN ' 209                                          : REM ENTRY
8012  SELECT  PRINT 005                                    : REM SEL OUTPUT DISPLY
8016  PRINT   HEX(03)                                      : REM CLR CRT SCREEN
8020  PRINT "  Command                When to Use"
8024  PRINT "  -------                -----------"
8028  PRINT " ASSIGN          This command must be the first command."
8032  PRINT "                 Purpose to assign data and program disks"
8036  PRINT "                 to the Phase II programs."
8040  PRINT
8044  PRINT " INITIAL         This command will initialize the Phase II"
8048  PRINT "                 Data disk before you beging data entry."
8052  PRINT
8056  PRINT " MAINTENANCE     This command will ask for the password"
8060  PRINT "                 for the Phase II Data Disk before"
8064  PRINT "                 any processing can be performed."
8068  PRINT
8072  PRINT " BACKUP          BACKUP will copy the contents of an"
8076  PRINT "                 existing Data Base onto a Backup Disk."
8080  REM **
8084  SELECT PRINT 405                                     : REM SEL OUTPUT DISPLY
8088  GOSUB ' 208 (16)                                     : REM RESPONCE WAIT
8092  RETURN                                               : REM EXIT
```

```
8096 REM **
8100 REM **
8104  REM **  Subroutine  ********************************************** 2 1 0
8108  REM **  Abstract >
8112  REM **    To Input the device structure to be assigned.
8116  REM **  Variables >
8120  REM **    R2  > input numeric val ! R1 > device tbl indx
8124  REM **  Device Specifications >
8128  REM **    #1  > AUTHOR Source Programs
8132  REM **    #2  > AUTHOR Phase II Data Base
8136  REM **    #3  > AUTHOR Phase II Backup Data Base
8140  REM **
8144  REM ************************************************
8148  DEFFN ' 210                                          : REM ENTRY
8152  DIM           W2$(5)2,W6$(5,2)1                      : REM CMMND TABLE
8156  IF            W6$(5,1) = HEX(00)       THEN 8168     : REM CHCK FLG
8160  INIT(20)      W6$()                                  : REM CLR ITEM FLAGS
8164  GOTO          8172                                   : REM DISPLAY MENU
8168                W6$(5,1) = HEX(20)                     : REM CLR FLG
8172                W2$(1) = "PR"                          : REM INIT TABLE
8176                W2$(2) = "DA"                          : REM INIT TABLE
8180                W2$(3) = "BA"                          : REM INIT TABLE
8184                W2$(4) = HEX(FFFF)                     : REM INIT TABLE
8188                W2$(5) = "EX"                          : REM INIT TABLE
8192  SELECT        PRINT 005(80)                          : REM SEL OUT DISPLY
8196  PRINT         HEX(03)                                : REM CLR CRT SCREEN
8200  PRINTUSING    8436, " "                              : REM PRINT MENU
8204  PRINTUSING    8440, " "                              : REM PRINT MENU
8208  PRINTUSING    8444, " "                              : REM PRINT MENU
8212  PRINTUSING    8448, W6$(1,1),W6$(1,2)                : REM PRINT MENU
8216  PRINTUSING    8452, W6$(2,1),W6$(2,2)                : REM PRINT MENU
8220  PRINTUSING    8456, W6$(3,1),W6$(3,2)                : REM PRINT MENU
8224  PRINTUSING    8460, " "                              : REM PRINT MENU
8228  SELECT        PRINT 405                              : REM SEL OUT DSPLY
8232  IF            W6$(5,1) <> HEX(20)      THEN 8324     : REM CHCK FLAG
8236  GOSUB ' 101   (15,1,1,0)                             : REM POSITION CUSOR
8240  PRINT         "Enter Command > "                     : REM PRNT PROMT
8244  GOSUB ' 104   (8,"A","EXIT")                         : REM INPUT COMMAND
8248                R0 = LEN(R2$)                          : REM CMPUT STRNG LEN
8252  IF            R0 <= 2                  THEN 8260     : REM WITHEN LIMITS
8256                R0  = 2                                : REM SET TO MAX LEN
8260  AND           (STR(R2$,1,R0),DF)                     : REM FORCE UPPR CASE
8264  MAT SEARCH    W2$(),=STR(R2$,1,R0) TO R2$() STEP 2:  REM SRCH CMND TBL
8268  IF            R2$(1) <> HEX(0000)      THEN 8292     : REM CMND FOUND
8272  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
8276  PRINT         "INVALID COMMAND      RE-ENTER"        : REM PRNT ERR MSG
8280  $GIO          PAUSE TONE/005(R8$(4),R8$(9))          : REM PAUSE TONE
8284  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
8288  GOTO          8236                                   : REM RE-ENTER CMND
8292                R0 = INT(FNR(VAL(R2$(1)))/2) + 1       : REM CMPUT DEV TYPE
```

```
8296  REM **
8300  REM ********************************************** INPUT DEV NUMBER
8304  REM **
8308                W6$(R0,1) = HEX(2A)                          : REM SET ITEM FLAG
8312  IF            W6$(5,1) <> HEX(20)       THEN 8324          : REM CHECK FLAG
8316                W6$(5,1)  = HEX(00)                          : REM SET FLAG
8320  GOTO          8192                                         : REM REPRINT MENU
8324  IF            R0 <> 5                   THEN 8340          : REM BRNCH TO EXIT
8328                R0  = 4                                      : REM SET TO EXIT IDX
8332                R1 =  6                                      : REM SET TO NULL BRNCH
8336  GOTO          8464                                         : REM BRANCH TO EXIT
8340  GOSUB ' 101   (10,1,1,0)                                   : REM POSITION CUSOR
8344  SELECT        PRINT 005(64)                                : REM SEL OUTPUT DISPLY
8348  PRINT         "  1 - Flopy Disk Drive #1    <310>"
8352  PRINT         "  2 - Flopy Disk Drive #2    <B10>"
8356  PRINT         "  3 - Flopy Disk Drive #3    <350>"
8360  PRINT         "  4 - Fixed Disk Plater      <320>"
8364  PRINT         "  5 - Removable Disk Plater  <B20>"
8368  SELECT        PRINT 405                                    : REM SEL OUTPUT DISPLY
8372  GOSUB ' 101   (16,1,1,0)
8376  PRINT         "Enter Disk Device to Assign > "
8380  CONVERT       R0 TO R0$(1), (#)                            : REM CMPUT DEFLT DEV
8384  GOSUB ' 104   (1,"N",R0$(1))                               : REM INPUT DEV #
8388  IF            R2 > 5                    THEN 8396          : REM INVALID INPUT
8392  IF            R2 > 0                    THEN 8416          : REM VALID INPUT
8396  GOSUB ' 101   (16,1,1,0)                                   : REM POSITION CUSOR
8400  PRINT         "INVALID COMMAND     RE-ENTER"               : REM PRNT ERR MSG
8404  $GIO          PAUSE TONE/005(R8$(4),R8$(9))                : REM WARNING TONE
8408  GOSUB ' 101   (16,1,1,0)                                   : REM POSITION CUSOR
8412  GOTO          8372                                         : REM RE-ENTER CMND
8416                R1 = R2                                      : REM STOR DEV #
8420                R2 = R0                                      : REM STOR DEV TYPE
8424  CONVERT       R1 TO W6$(R0,2), (#)                         : REM STOR DEV ASSIND
8428  GOSUB ' 101   (10,1,7,0)                                   : REM ERASE DISK MENU
8432  REM                                                        : REM IMAGE FORMATS
8436 % Command Summary; #
8440 %     Command      Device Assigned      Function #
8444 % # ---------    -----------------    ---------------------------
8448 %   # PROGRAM       Disk; #           Assign the Program Disk
8452 %   # DATA          Disk; #           Assign INPUT Data Base Disk
8456 %   # BACKUP        Disk; #           Assign BACKUP Data Base Disk
8460 %    -EXIT              #            Return to Command Menu
8464  RETURN                                                     : REM EXIT
```

```
8468  REM **
8472  REM **
8476  REM **  Subroutine  ********************************************** 2 1 1
8480  REM **  Abstract >
8484  REM **    To assign device to internal table slot.
8488  REM **  Variables >
8492  REM **    R2 > device type     ! R1 > device number
8496  REM **
8500  REM ********************************************
8504  DEFFN ' 211                                              : REM ENTRY
8508 REM **
8512 REM **          ! 310  ! B10  ! 350  ! 320  ! B20  ! EXIT !
8516 REM **          !------!------!------!------!------!------!
8520  ON R1 GOTO     8532  , 8552 , 8572 , 8592 , 8612 , 8632 : REM BRANCH
8524 REM **          !------!------!------!------!------!------!
8528 REM **
8532  ON R2 GOTO    8536, 8540, 8544                           : REM BRNCH TO DEV
8536  SELECT        #1 310                    : GOTO 8632      : REM ASSIGN #1
8540  SELECT        #2 310                    : GOTO 8632      : REM ASSIGN #2
8544  SELECT        #3 310                    : GOTO 8632      : REM ASSIGN #3
8548 REM **
8552  ON R2 GOTO    8556, 8560, 8564                           : REM BRNCH TO DEV
8556  SELECT        #1 B10                    : GOTO 8632      : REM ASSIGN #1
8560  SELECT        #2 B10                    : GOTO 8632      : REM ASSIGN #2
8564  SELECT        #3 B10                    : GOTO 8632      : REM ASSIGN #3
8568 REM **
8572  ON R2 GOTO    8576, 8580, 8584                           : REM BRNCH TO DEV
8576  SELECT        #1 350                    : GOTO 8632      : REM ASSIGN #1
8580  SELECT        #2 350                    : GOTO 8632      : REM ASSIGN #2
8584  SELECT        #3 350                    : GOTO 8632      : REM ASSIGN #3
8588 REM **
8592  ON R2 GOTO    8596, 8600, 8604                           : REM BRNCH TO DEV
8596  SELECT        #1 320                    : GOTO 8632      : REM ASSIGN #1
8600  SELECT        #2 320                    : GOTO 8632      : REM ASSIGN #2
8604  SELECT        #3 320                    : GOTO 8632      : REM ASSIGN #3
8608 REM **
8612  ON R2 GOTO    8616, 8620, 8624                           : REM BRNCH TO DEV
8616  SELECT        #1 B20                    : GOTO 8632      : REM ASSIGN #1
8620  SELECT        #2 B20                    : GOTO 8632      : REM ASSIGN #2
8624  SELECT        #3 B20                    : GOTO 8632      : REM ASSIGN #3
8628 REM **
8632  RETURN                                                   : REM EXIT
```

AUTHOR System: Phase 2 - ATH2INT2 - # MAP

```
7084  - 7112
7120  - 7096
7200  - 7052
7252  - 7304
7312  - 7284
7340  - 7324
7344  - 7332
7396  - 7436
7444  - 7408
7456  - 7496
7504  - 7472 7476
7588  - 7512
7596  - 7504
7632  - 7684
7692  - 7676
7728  - 7708
7812  - 7820 7824
7936  - 7932
7940  - 7932
7948  - 7936
8168  - 8156
8172  - 8164
8192  - 8320
8236  - 8288
8260  - 8252
8292  - 8268
8324  - 8232 8312
8340  - 8324
8372  - 8412
8396  - 8388
8416  - 8392
8436  - 8200
8440  - 8204
8444  - 8208
8448  - 8212
8452  - 8216
8456  - 8220
8460  - 8224
8464  - 8336
8532  - 8520
8536  - 8532
8540  - 8532
8544  - 8532
8552  - 8520
8556  - 8552
8560  - 8552
8564  - 8552
8572  - 8520
8576  - 8572
8580  - 8572
8584  - 8572
8592  - 8520
8596  - 8592
8600  - 8592
8604  - 8592
8612  - 8520
8616  - 8612
8620  - 8612
```

8624 - 8612
8632 - 8520 8536 8540 8544 8556 8560 8564 8576 8580 8584 8596 8600 8604 8616
8620 8624

AUTHOR System: Phase 2 - ATH21NT2 - V MAP

```
B1$(   - 7636 7740 7744
H8$(   - 7448
J0     - 7900 7904 7936 7956
J0$    - 7888 7904 7928
J1     - 7908 7912 7936 7952
J1$    - 7888 7912 7928
J2     - 7916 7920 7936 7948
J2$    - 7888 7920 7928
J3     - 7940 7944
J3$    - 7888
K1     - 7884 7892 7924 7960
Q5     - 7516 7532 7536 7544 7556
Q6     - 7532 7536 7540
Q6$(   - 7204 7540
Q7     - 7536 7540
Q8$(   - 7120 7132 7152 7172 7176 7180 7200 7220 7244 7316 7328 7340 7524 7540
Q9$(   - 7092 7096 7404 7408 7444 7448 7468
R0     - 7052 7324 7504 7692 7712 7720 7736 7820 8248 8252 8256 8260 8264 8292
         8308 8324 8328 8380 8420 8424
R0$(   - 7932 8380 8384
R1     - 7128 7132 7136 7148 7152 7160 7164 7404 7520 7524 7564 7824 8332 8416
         8424 8520
R2     - 7092 7152 7180 7328 7340 7524 7528 7540 7560 7736 7740 7744 7748 7760
         8388 8392 8416 8420 8532 8552 8572 8592 8612
R2$    - 7240 7264 7284 7312 7472 7476 7544 7548 7552 7676 7708 8248 8260 8264
R2$(   - 7124 7132 8264 8268 8292
R3     - 7740 7744
R3$    - 7264 7284
R8$(   - 7108 7296 7420 7488 7756 8280 8404
R9$(   - 7208 7212 7216 7220 7240 7244 7312 7316 7444 7452 7468 7472 7748 7752
         7756
W2$(   - 8152 8172 8176 8180 8184 8188 8264
W6$(   - 8152 8156 8160 8168 8212 8216 8220 8232 8308 8312 8316 8424
```

```
???? DEFFN' 101
       - 7100 7140 7156 7252 7272 7288 7300 7412 7456 7480 7492 7508 7664 7696
         7728 7892 7924 7960 8236 8272 8284 8340 8372 8396 8408 8428
???? DEFFN' 104
       - 7236 7260 7280 7464 7672 7704 8244 8384
???? DEFFN' 135
       - 7184 7572
???? DEFFN' 202
       - 7428
???? DEFFN' 203
       - 7432
7048 DEFFN' 204
       - 7716
7396 DEFFN' 205
       - 7640
7632 DEFFN' 206
7808 DEFFN' 207
       - 7680
7884 DEFFN' 208
       - 7084 7424 8088
8008 DEFFN' 209
8148 DEFFN' 210
       - 7812
8504 DEFFN' 211
       - 7816
```

4.4 ATH2LOGC

The purpose of the 'ATH2LOGC' program is to provide the control logic for executing the data entry commands. The 'ATH2LOGC' program is presented here in two parts.

4.4.1 ATH2LOG1

This section of the 'ATH2LOGC' program contains the main branching instructions for the execution of the data entry commands. This section also contains subroutines 201 to 217.

The remainder of this section presents the 'ATH2LOGI' program listing and reference maps.

```
6000  %##  Program > AUTHOR/2  (ATH2LOG1)  Version > 2.0 2200 T/VP  780320
6004  REM **  Abstract  ***************************************************
6008  REM **                                                              **
6012  REM **    ATH2LOGC will provide the logic for                       **
6016  REM **    executing the commands >                                  **
6020  REM **       TITLE        ORGANIZATION        TYPE          SYMBOL.  **
6024  REM **       SUBTITLE     TRAINING            NAME          HELP     **
6028  REM **       AUTHORS      IMPORTANCE          SOURCE        LIST     **
6032  REM **       DATE         USAGE               EXAMPLE       ESCAPE   **
6036  REM **                                                              **
6040  REM ***************************************************************
6044  REM **
6048  GOSUB ' 105 (R0,"Data Entry Procedures Menu")     : REM INPUT COMMAND
6052  PRINT       HEX(03)                               : REM CLR CRT SCREEN
6056  GOSUB ' 101 (14,1,0,0)                            : REM POSITION CUSOR
6060 REM **
6064 REM **         ! TITLE  ! SUBTITL ! AUTHOR ! DATE ! ORGAN !
6068 REM **         !--------!---------!--------!------!-------!
6072  ON R0-0  GOTO  6120   , 6124    , 6128   , 6132 , 6136  ,
                     6140   , 6144    , 6148   , 6152 , 6156  : REM BRANCH
6076 REM **         !--------!---------!--------!------!-------!
6080 REM **         ! TRAIN  ! IMPORTN ! USAGE  ! TYPE ! NAME  !
6084 REM **
6088 REM **
6092 REM **         ! SOURCE ! EXAMPLE ! SYMBOL ! HELP ! LIST  !
6096 REM **         !--------!---------!--------!------!-------!
6100  ON R0-10 GOTO 6160    , 6164    , 6168   , 6172 , 6176  ,
                     6180   , 6184    , 6184   , 6184 , 6184  : REM BRANCH
6104 REM **         !--------!---------!--------!------!-------!
6108 REM **         ! ESCAPE ! NULL    ! NULL   ! NULL ! NULL  !
6112 REM **
6116  REM
6120  GOSUB ' 201                          : GOTO 6048 : REM TITLE
6124  GOSUB ' 213                          : GOTO 6048 : REM SUBTITLE
6128  GOSUB ' 214                          : GOTO 6048 : REM AUTHORS
6132  GOSUB ' 215                          : GOTO 6048 : REM DATE
6136  GOSUB ' 216                          : GOTO 6048 : REM ORGANIZATION
6140  GOSUB ' 202                          : GOTO 6048 : REM TRAINING
6144  GOSUB ' 203                          : GOTO 6048 : REM IMPORTANCE
6148  GOSUB ' 204                          : GOTO 6048 : REM USAGE
6152  GOSUB ' 206                          : GOTO 6048 : REM TYPE
6156  GOSUB ' 207                          : GOTO 6048 : REM NAME
6160  GOSUB ' 208                          : GOTO 6048 : REM SOURCE
6164  GOSUB ' 205                          : GOTO 6048 : REM EXAMPLE
6168  GOSUB ' 209                          : GOTO 6048 : REM SYMBOL
6172  GOSUB ' 211                          : GOTO 6048 : REM HELP
6176  GOSUB ' 217                          : GOTO 6048 : REM LIST
6180  GOSUB ' 212                          : GOTO 6048 : REM ESCAPE
6184  GOSUB ' 211                          : GOTO 6048 : REM NULL CMND
6188 REM **
```

```
6192 REM **
6196 REM **                                                    LABLED SUBROUTINES
6200 REM **
6204  REM **  Subroutine  ********************************************** 2 0 1
6208  REM **  Command   >  TITLE
6212  REM **  Function  >
6216  REM **    To allow the entry of the Title.
6220  REM **  Variables >
6224  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6228  REM **
6232  REM ************************************************
6236  DEFFN ' 201                                          : REM ENTRY
6240  INIT(20)     R9$()                                   : REM CREATE DASH LINE
6244  $CIO DASH    /005 (R8$(2),R8$(9))R9$()<1,63>         : REM PRINT LINE
6248  PRINT        "Enter Title."; HEX(0D0A)               : REM PRINT PROMPT
6252  PRINT        "Touch 'ESCAPE' key when Completed."    : REM PRINT INSTRUCTN
6256  GOSUB ' 106 (00001,1)                                : REM ENTER & SAVE TEXT
6260              N1$ = "SUBTITLE"                        : REM SET NXT COMMAND
6264  RETURN                                               : REM EXIT
```

```
6268 REM **
6272 REM **
6276  REM **  Subroutine  ********************************************** 2 0 2
6280  REM **  Command   >  TRAINING
6284  REM **  Function  >
6288  REM **    To allow the entry of the Training Objective.
6292  REM **  Variables >
6296  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6300  REM **
6304  REM ***********************************************
6308  DEFFN ' 202                                          : REM ENTRY
6312  INIT(20)     R9$()                                   : REM CREATE DASH LINE
6316  $GID DASH    /005 (R8$(2),R8$(9))R9$()<1,63>         : REM PRINT LINE
6320  PRINT        "Enter Training Objective.";HEX(0D0A): REM PRINT PROMPT
6324  PRINT        "Touch 'ESCAPE' key when Completed." : REM PRINT INSTRUCTN
6328  GOSUB ' 106 (00011,1)                             : REM ENTER & SAVE TEXT
6332               N1$ = "IMPORTANCE"                   : REM SET NXT COMMAND
6336  RETURN                                            : REM EXIT
```

```
6340 REM **
6344 REM **
6348  REM **  Subroutine  ********************************************** 2 0 3
6352  REM **  Command   >  IMPORTANCE
6356  REM **  Function  >
6360  REM **    To allow the entry of the Importance of
6364  REM **    the symbols.
6368  REM **  Variables >
6372  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6376  REM **
6380  REM ************************************************
6384  DEFFN ' 203                                          : REM ENTRY
6388  INIT(20)     R9$()                                   : REM CREATE DASH LINE
6392  $GIU DASH    /005 (R8$(2),R8$(9))R9$()<1,63>         : REM PRINT LINE
6396  PRINT        "Enter the Importance of the Symbols.";HEX(0D0A)
6400  PRINT        "Touch 'ESCAPE' key when Completed."    : REM PRINT INSTRUCTN
6404  GOSUB ' 106 (00031,1)                                : REM ENTER & SAVE TEXT
6408               N1$ = "USAGE"                           : REM SET NXT COMMAND
6412  RETURN                                               : REM EXIT
```

```
6416 REM **
6420 REM **
6424  REM **  Subroutine  ********************************************** 2 0 4
6428  REM **  Command    >  USAGE
6432  REM **  Function   >
6436  REM **    To allow the entry of How the Symbols will
6440  REM **    be Used.
6444  REM **  Variables >
6448  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6452  REM **
6456  REM ***********************************************
6460  DEFFN ' 204                                          : REM ENTRY
6464  INIT(20)     R9$()                                   : REM CREATE DASH LINE
6468  $GIO DASH    /005 (R8$(2),R8$(9))R9$()<1,63>         : REM PRINT LINE
6472  PRINT        "Enter How Symbol to be Used.";HEX(0D0A)
6476  PRINT        "Touch 'ESCAPE' key when Completed."    : REM PRINT INSTRUCTN
6480  GOSUB ' 106 (00032,1)                                : REM ENTER & SAVE TEXT
6484               N1$ = "TYPE"                            : REM SET NXT COMMAND
6488  RETURN                                               : REM EXIT
```

```
6492 REM **
6496 REM **
6500  REM **  Subroutine  ********************************************** 2 0 5
6504  REM **  Command   >  EXAMPLE
6508  REM **  Function  >
6512  REM **    To allow the entry of an Example of
6516  REM **    the symbols.
6520  REM **  Variables >
6524  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6528  REM **
6532  REM ***********************************************
6536  DEFFN ' 205                                          : REM ENTRY
6540  INIT(2D)    R9$()                                    : REM CREATE DASH LINE
6544  $CIO DASH   /005 (R8$(2),R8$(9))R9$()<1,63>          : REM PRINT LINE
6548  PRINT       "Enter the Example.";HEX(0D0A)
6552  PRINT       "Touch 'ESCAPE' key when Completed."     : REM PRINT INSTRUCTN
6556  GOSUB ' 106 (00041,1)                                : REM ENTER & SAVE TEXT
6560              N1$ = "SYMBOL"                           : REM SET NXT COMMAND
6564  RETURN                                               : REM EXIT
```

```
6568 REM **
6572 REM **
6576  REM **  Subroutine  ********************************************** 2 0 6
6580  REM **  Command   >  TYPE
6584  REM **  Function  >
6588  REM **    To allow the entry of the Type of Symbol.
6592  REM **  Variables >
6596  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6600  REM **
6604  REM ***********************************************
6608  DEFFN ' 206                                          : REM ENTRY
6612  INIT(2D)    R9$()                                    : REM CREATE DASH LINE
6616  $GIO DASH   /005 (R8$(2),R8$(9))R9$()<1,63>          : REM PRINT LINE
6620  PRINT       "Enter Type of Symbol.";HEX(0D0A)
6624  PRINT       "Touch 'ESCAPE' key when Completed."     : REM PRINT INSTRUCTN
6628  GOSUB ' 106 (00022,1)                                : REM ENTER & SAVE TEXT
6632              N1$ = "NAME"                             : REM SET NXT COMMAND
6636  RETURN                                               : REM EXIT
```

```
6640 REM **
6644 REM **
6648  REM **  Subroutine  ********************************************** 2 0 7
6652  REM **  Command   >  NAME
6656  REM **  Function  >
6660  REM **    To allow the entry of the Name of the Users.
6664  REM **  Variables >
6668  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6672  REM **
6676  REM **********************************************
6680  DEFFN ' 207                                          : REM ENTRY
6684  INIT(2D)     R9$()                                   : REM CREATE DASH LINE
6688  $GIO DASH    /005 (R8$(2),R8$(9))R9$()<1,63>         : REM PRINT LINE
6692  PRINT        "Enter the Name of the Users.";HEX(0D0A)
6696  PRINT        "Touch 'ESCAPE' key when Completed."    : REM PRINT INSTRUCTN
6700  GOSUB ' 106 (00021,1)                                : REM ENTER & SAVE TEXT
6704               N1$ = "SOURCE"                          : REM SET NXT COMMAND
6708  RETURN                                               : REM EXIT
```

```
6712 REM **
6716 REM **
6720  REM **  Subroutine  ********************************************** 2 0 8
6724  REM **  Command    >  SOURCE
6728  REM **  Function   >
6732  REM **    To allow the entry of the Offical Source of
6736  REM **    Reference for definition of symbols.
6740  REM **  Variables >
6744  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6748  REM **
6752  REM ************************************************
6756  DEFFN ' 208                                          : REM ENTRY
6760  INIT(20)     R9$()                                   : REM CREATE DASH LINE
6764  $GIO DASH    /005 (R8$(2),R8$(9))R9$()<1,63>         : REM PRINT LINE
6768  PRINT        "Enter Official Source of Reference.";HEX(0D0A)
6772  PRINT        "Touch 'ESCAPE' key when Completed."    : REM PRINT INSTRUCTN
6776  GOSUB ' 106 (00023,1)                                : REM ENTER & SAVE TEXT
6780               N1$ = "EXAMPLE"                         : REM SET NXT COMMAND
6784  RETURN                                               : REM EXIT
```

```
6788 REM **
6792 REM **
6796  REM **  Subroutine  ********************************************** 209
6800  REM **  Command   >  SYMBOL
6804  REM **  Function  >
6808  REM **    To allow the entry of Symbol Descriptions.
6812  REM **  Variables >
6816  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
6820  REM **    L1    > set number      ! M1$   > set number
6824  REM **    L2    > symbol number   ! M2$   > symbol number
6828  REM **    L3    > item number     ! M3$   > set name
6832  REM **    L4    > graphic code    ! M4$   > graphic code
6836  REM **    R2    > input numric val! R2$   > inpt alph val
6840  REM **    K1    > crt row indx    ! K2    > crt col indx
6844  REM **    K5    > buffer indx     ! K6    > line number
6848  REM **  Function Definitions >
6852  REM **    I(L3) > compute item ID for index lookup.
6856  REM **
6860  REM **********************************************
6864  DEFFN ' 209                                          : REM ENTRY
6868  DIM         M1$3,M2$4,M3$40,M4$1                     : REM DEFINE BUFRS
6872  DEFFN       I(L3) = L1*1000 + L2*10 + L3             : REM DEFINE FUNCTION
6876              M1$   = "001"                            : REM INIT SBL SET #
6880              M2$   = "0001"                           : REM INIT SBL NUMBER
6884              M3$   = "Symbol Set #xxx"                : REM INIT SBL SET NAM
6888              M4$   = "1"                              : REM INIT GRAPHIC CODE
6892              N2$   = "SETNUMBER"                      : REM INIT FIRST ITEM
6896  INIT(20)    Q4$()                                    : REM CLR ITEM FLG BUFR
6900              L1    = 0                                : REM INIT SET # INDX
6904              L2    = 0                                : REM INIT SBL # INDX
6908  GOSUB ' 230 (R0)                                     : REM INPUT COMMAND
6912 REM **       ! S NUM  ! SB NUM ! S NAME ! RF NUM  ! SYMBOL !
6916 REM **       !--------!-------!--------!---------!--------!
6920  ON R0 GOTO    6944   , 6948   , 6952   , 6956    , 6964   ,
                    6968   , 6972   , 6976   , 6980    , 6984   : REM BRANCH
6924 REM **       !--------!-------!--------!---------!--------!
6928 REM **       ! DEFFN  ! MEMAID ! SIMSBL ! NEXT    ! DELETE !
6932 REM **
6936  GOTO        6992                                     : REM BRANCH TO EXIT
6940 REM **
6944  GOSUB ' 218                          : GOTO 6908 : REM ENTER SET NUMBER
6948  GOSUB ' 219                          : GOTO 6908 : REM ENTER SBL NUMBR
6952  GOSUB ' 220                          : GOTO 6908 : REM ENTER SET NAME
6956  GOSUB ' 221                                      : REM CONSTURCT REF #
6960  GOSUB ' 222                          : GOTO 6908 : REM SAVE SET,SBL #
6964  GOSUB ' 223                          : GOTO 6908 : REM ENTER SYMBOL
6968  GOSUB ' 224                          : GOTO 6908 : REM ENTER DEFINITION
6972  GOSUB ' 225                          : GOTO 6908 : REM ENTER MNEMONIC
6976  GOSUB ' 226                          : GOTO 6908 : REM ENTER SIM SMBLS
6980  GOSUB ' 227                          : GOTO 6908 : REM INCR COUNTERS
6984  GOSUB ' 228                          : GOTO 6900 : REM DELETE CRNT SBL
6988 REM **
6992  RETURN                                           : REM EXIT
```

```
AUTHOR System: Phase 2 - ATH2LOG1 - 2 1 1

6996 REM **
7000 REM **
7004  REM **  Subroutine  ********************************************** 2 1 1
7008  REM **
7012  REM **  Command   >  HELP
7016  REM **  Function  >
7020  REM **    To Re-Display the summary of Commands.
7024  REM **
7028  REM ********************************************
7032  DEFFN ' 211                                              : REM ENTRY
7036  RETURN                                                   : REM EXIT
```

```
7040 REM **
7044 REM **
7048  REM **  Subroutine  ********************************************** 2 1 2
7052  REM **  Command   >  ESCAPE
7056  REM **  Function  >
7060  REM **    To Terminate program execution.
7064  REM **
7068  REM ************************************************
7072  DEFFN ' 212                                          : REM ENTRY
7076  PRINT    HEX(0D)
7080  PRINT    "A U T H O R   S Y S T E M";HEX(0D0A)
7084  PRINT    "      P H A S E   II"
7088  GOSUB '  101 (15,1,0,0)
7092  PRINT    "AUTHOR SYSTEM PHASE II";HEX(0D0C0C)
7096  PRINT    "END PROGRAM EXECUTION";HEX(0D0C)
7100  COM      CLEAR 00                                    : REM CLEAR COMMON TBL
7104  LOAD DC  T #1, "AUTHOR"
7108  END                                                  : REM END OF PROGRAM
```

```
7112 REM **
7116 REM **
7120  REM **  Subroutine  ********************************************** 2 1 3
7124  REM **  Command   >  SUBTITLE
7128  REM **  Function  >
7132  REM **    To allow the entry of the Subtitle.
7136  REM **  Variables >
7140  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
7144  REM **
7148  REM ***********************************************
7152  DEFFN ' 213                                          : REM ENTRY
7156  INIT(20)     R9$()                                   : REM CREATE DASH LINE
7160  $GIO DASK    /005 (R8$(2),R8$(9)) R9$()<1,63>        : REM PRINT LINE
7164  PRINT        "Enter the Subtitle.";HEX(000A)         : REM PRINT INSTRUCTION
7168  PRINT        "Touch 'ESCAPE' key when Completed.":   REM PRINT INSTRUCTION
7172  GOSUB ' 106 (00002,1)                                : REM ENTER & SAVE TEXT
7176              N1$ = "AUTHORS"                          : REM SET NEXT COMMAND
7180  RETURN                                               : REM EXIT
```

```
7184 REM **
7188 REM **
7192  REM **  Subroutine  ********************************************** 2 1 4
7196  REM **  Command   >  AUTHORS
7200  REM **  Function  >
7204  REM **    To allow the entry of the Author(s).
7208  REM **  Variables >
7212  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
7216  REM **
7220  REM **********************************************
7224  DEFFN ' 214                                          : REM ENTRY
7228  INIT(2D)     R9$()                                   : REM CREATE DASH LINE
7232  $GIO DASK    /005 (R8$(2),R8$(9)) R9$()<1,63>        : REM PRINT LINE
7236  PRINT        "Enter the Author's.";HEX(0D0A)         : REM PRINT INSTRUCTION
7240  PRINT        "Touch 'ESCAPE' key when Completed.":   REM PRINT INSTRUCTION
7244  GOSUB ' 106 (00003,1)                                : REM ENTER & SAVE TEXT
7248                N1$ = "DATE"                           : REM SET NEXT COMMAND
7252  RETURN                                               : REM EXIT
```

```
7256 REM **
7260 REM **
7264  REM **  Subroutine  ********************************************** 2 1 5
7268  REM **  Command   >  DATE
7272  REM **  Function  >
7276  REM **    To allow the entry of the Date.
7280  REM **  Variables >
7284  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
7288  REM **
7292  REM ***********************************************
7296  DEFFN ' 215                                          : REM ENTRY
7300  INIT(2D)     R9$()                                   : REM CREATE DASH LINE
7304  $GIO DASH    /005 (R8$(2),R8$(9)) R9$()<1,63>        : REM PRINT LINE
7308  PRINT        "Enter the Date      ";HEX(0D0A)        : REM PRINT INSTRUCTION
7312  PRINT        "Touch 'ESCAPE' key when Completed.":   REM PRINT INSTRUCTION
7316  GOSUB ' 106 (00004,1)                                : REM ENTER & SAVE TEXT
7320                N1$ = "ORGANIZATION"                   : REM SET NEXT COMMAND
7324  RETURN                                               : REM EXIT
```

```
7328 REM **
7332 REM **
7336  REM **  Subroutine  ********************************************** 2 1 6
7340  REM **  Command   >  ORGANIZATION
7344  REM **  Function  >
7348  REM **    To allow the entry of the Name of the
7352  REM **    Organization.
7356  REM **  Variables >
7360  REM **    R8$() > micro cmnd bufr ! R9$() > scratch bufr
7364  REM **
7368  REM ***********************************************
7372  DEFFN ' 216                                          : REM ENTRY
7376  INIT(2D)     R9$()                                   : REM CREATE DASH LINE
7380  $GIO DASH    /005 (R8$(2),R8$(9)) R9$()<1,63>        : REM PRINT LINE
7384  PRINT        "Enter the Name of the Organization.";HEX(0D0A)
7388  PRINT        "Touch 'ESCAPE' key when Completed.": REM PRINT INSTRUCTION
7392  GOSUB ' 106 (00005,1)                                : REM ENTER & SAVE TEXT
7396                N1$ = "TRAINING"                       : REM SET NEXT COMMAND
7400  RETURN                                               : REM EXIT
```

```
7404 REM **
7408 REM **
7412  REM **  Subroutine  ********************************************** 2 1 7
7416  REM **  Command   >  LIST
7420  REM **  Function  >
7424  REM **    To Load List processing Routine.
7428  REM **
7432  REM ***********************************************
7436  DEFFN ' 217                                          : REM ENTRY
7440  GOSUB ' 003                                          : REM LOAD LISTER
7444  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 2 - ATH2LOG1 - # MAP

```
(0010) - 7104
 6048  - 6120 6124 6128 6132 6136 6140 6144 6148 6152 6156 6160 6164 6168 6172
         6176 6180 6184
 6120  - 6072
 6124  - 6072
 6128  - 6072
 6132  - 6072
 6136  - 6072
 6140  - 6072
 6144  - 6072
 6148  - 6072
 6152  - 6072
 6156  - 6072
 6160  - 6100
 6164  - 6100
 6168  - 6100
 6172  - 6100
 6176  - 6100
 6180  - 6100
 6184  - 6100
 6900  - 6984
 6908  - 6944 6948 6952 6960 6964 6968 6972 6976 6980
 6944  - 6920
 6948  - 6920
 6952  - 6920
 6956  - 6920
 6964  - 6920
 6968  - 6920
 6972  - 6920
 6976  - 6920
 6980  - 6920
 6984  - 6920
 6992  - 6936
```

AUTHOR System: Phase 2 - ATH2LOG1 - V MAP

```
D0     - 7100
L1     - 6872 6900
L2     - 6872 6904
L3     - 6872
M1$    - 6868 6876
M2$    - 6868 6880
M3$    - 6868 6884
M4$    - 6868 6888
N1$    - 6260 6332 6408 6484 6560 6632 6704 6780 7176 7248 7320 7396
N2$    - 6892
Q4$(   - 6896
R0     - 6048 6072 6100 6908 6920
R8$(   - 6244 6316 6392 6468 6544 6616 6688 6764 7160 7232 7304 7380
R9$(   - 6240 6244 6312 6316 6388 6392 6464 6468 6540 6544 6612 6616 6684 6688
         6760 6764 7156 7160 7228 7232 7300 7304 7376 7380
```

```
AUTHOR System:  Phase 2 - ATH2LOG1 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
       - 7440
???? DEFFN' 101
       - 6056 7088
???? DEFFN' 105
       - 6048
???? DEFFN' 106
       - 6256 6328 6404 6480 6556 6628 6700 6776 7172 7244 7316 7392
6236 DEFFN' 201
       - 6120
6308 DEFFN' 202
       - 6140
6384 DEFFN' 203
       - 6144
6460 DEFFN' 204
       - 6148
6536 DEFFN' 205
       - 6164
6608 DEFFN' 206
       - 6152
6680 DEFFN' 207
       - 6156
6756 DEFFN' 208
       - 6160
6864 DEFFN' 209
       - 6168
7032 DEFFN' 211
       - 6172 6184
7072 DEFFN' 212
       - 6180
7152 DEFFN' 213
       - 6124
7224 DEFFN' 214
       - 6128
7296 DEFFN' 215
       - 6132
7372 DEFFN' 216
       - 6136
7436 DEFFN' 217
       - 6176
???? DEFFN' 218
       - 6944
???? DEFFN' 219
       - 6948
???? DEFFN' 220
       - 6952
???? DEFFN' 221
       - 6956
???? DEFFN' 222
       - 6960
???? DEFFN' 223
       - 6964
???? DEFFN' 224
       - 6968
???? DEFFN' 225
```

```
        - 6972
???? DEFFN' 226
        - 6976
???? DEFFN' 227
        - 6980
???? DEFFN' 228
        - 6984
???? DEFFN' 230
        - 6908
```

4.4.2 ATH2LOG2

This section of the 'ATH2LOGC' program contains subroutines 218 to 230.

The remainder of this section presents the 'ATH2LOG2' program listing and refe nce maps.

```
8000  %##  Program > AUTHOR/2  (ATH2LOG2)  Version > 2.0 2200 T/VP  780320
8004  REM **  Subroutine  ********************************************** 2 1 8
8008  REM **  Function >
8012  REM **    To allow the entry of the Set Number.
8016  REM **
8020  REM ***********************************************
8024  DEFFN ' 218                                                  : REM ENTRY
8028  PRINT       HEX(03)                                          : REM CLR CONSOL SCREN
8032  GOSUB ' 101 (14,1,0,0)                                       : REM POSITION CUSOR
8036  INIT(20)    R9$()                                            : REM CREATE DASH LINE
8040  $CIO DASH   /005 (R8$(2),R8$(9))R9$()<1,63>                  : REM PRINT LINE
8044  GOSUB ' 101 (1,1,0,0)                                        : REM POSITION CUSOR
8048  PRINT       "Enter Set Number > "                            : REM PRINT PROMPT
8052  GOSUB ' 104 (3,"N",M1$)                                      : REM INPUT SET NUMBR
8056              L1 = R2                                          : REM SAVE SET NUMBR
8060  CONVERT     R2 TO R2$, (###)                                 : REM CONV SET NUMBR
8064  GOSUB ' 101 (15,1,0,0)                                       : REM POSITION CUSOR
8068  PRINT       "Set # > ";STR(R2$,1,3)                          : REM PRINT SET NUMBR
8072  IF          STR(R2$,1,3) = M1$          THEN 8080            : REM SAVE SET NUMBR
8076              M2$ = "0001"                                     : REM RESET SBL NUMBR
8080              M1$ = R2$                                        : REM SAVE NEW SET #
8084              N2$ = "SBLNUMBER"                                : REM SET NEXT COMMAND
8088              Q4$(9) = HEX(20)                                 : REM CLR ITEM FLG-NEXT
8092  RETURN                                                       : REM EXIT
```

```
8096 REM **
8100 REM **
8104  REM **  Subroutine  ********************************************** 2 1 9
8108  REM **  Function >
8112  REM **    To allow the entry of the Symbol number.
8116  REM **
8120  REM ***********************************************
8124  DEFFN ' 219                                          : REM ENTRY
8128  GOSUB ' 101 (1,1,1,0)                                : REM POSITION CUSOR
8132  PRINT       "Enter Symbol Number > "                 : REM PRINT PROMPT
8136  GOSUB ' 104 (4,"N",M2$)                              : REM INPUT SBL NUMBR
8140              L2 = R2                                  : REM SAVE SBL NUMBR
8144  CONVERT     R2 TO M2$, (####)                        : REM CONV SBL NUMBR
8148  GOSUB ' 101 (16,1,0,0)                               : REM POSITION CUSOR
8152  PRINT       "Sbl # > ";M2$                           : REM PRINT SBL NUMBR
8156              N2$ = "SETNAME"                          : REM SET NEXT COMMAND
8160  RETURN                                               : REM EXIT
```

```
8164 REM **
8168 REM **
8172  REM **  Subroutine  ********************************************** 2 2 0
8176  REM **  Function >
8180  REM **    To allow the entry of the Set Name.
8184  REM **
8188  REM ***************************************************
8192  DEFFN ' 220                                              : REM ENTRY
8196  GOSUB ' 101 (1,1,1,0)                                    : REM POSITION CUSOR
8200  PRINT       "Enter Set Name > "                          : REM PRINT PROMPT
8204  GOSUB ' 104 (40,"A",M3$)                                 : REM INPUT SET NAME
8208  IF      STR(R2$,1,12) <> "Symbol Set #" THEN 8216:       REM NOT DEFAULT NAME
8212          STR(R2$,13,3) = M1$                              : REM INSRT SET NUMBR
8216          M3$   = R2$                                      : REM SAVE SET NAME
8220  GOSUB ' 101 (15,15,0,0)                                  : REM POSITION CUSOR
8224  PRINT   M3$                                              : REM PRINT SET NAME
8228          N2$ = "REFERENCE NUMBER"                         : REM SET NEXT COMMAND
8232  RETURN                                                   : REM EXIT
```

```
8236 REM **
8240 REM **
8244  REM **  Subroutine  ********************************************** 2 2 1
8248  REM **  Function >
8252  REM **    To construct the symbol reference number.
8256  REM **
8260  REM ***********************************************
8264  DEFFN ' 221                                          : REM ENTRY
8268  GOSUB ' 101 (1,1,1,0)                                : REM POSITION CUSOR
8272  PRINT   "Enter Graphic Code (1-Alpha, 2-Graphic) > "
8276  GOSUB ' 104 (1,"N",M4$)                              : REM INPUT SBL TYPE
8280  IF          R2 < 3                        THEN 8288  : REM VALID INPUT
8284              R2 = 1                                   : REM SET TO ALPHA TYPE
8288  CONVERT     R2 TO M4$, (#)                           : REM CONV TO DISPLAY
8292  GOSUB ' 101 (16,15,0,0)                              : REM POSITION CUSOR
8296  PRINT       "Graphic Code > ";M4$                    : REM PRINT VALUE
8300  GOSUB ' 101 (16,39,0,0)                              : REM POSITION CUSOR
8304  PRINTUSING  8308, "#",M4$,M1$,M2$,"00";              : REM PRINT REF NUMBR
8308  %Ref #  > # ### #### ##
8312              N2$ = "SYMBOL"                           : REM SET NEXT COMMAND
8316  RETURN                                               : REM EXIT
```

```
8320 REM **
8324 REM **
8328  REM **  Subroutine  ********************************************** 2 2 2
8332  REM **  Function >
8336  REM **    To store entered data on disk.
8340  REM **
8344  REM ***********************************************
8348  DEFFN ' 222                                                    : REM ENTRY
8352  INIT(20)    R1$()                                              : REM CLR STORAGE BUFR
8356  INIT(20)    R9$()                                              : REM CLR TEMP BUFR
8360              STR(R9$(1),1,1) = M4$                              : REM LOAD SBL TYPE
8364              STR(R9$(1),2,3) = M1$                              : REM LOAD SET NUMBR
8368              STR(R9$(1),5,4) = M2$                              : REM LOAD SBL NUMBR
8372              STR(R9$(1),9,1) = HEX(FF)                          : REM LOAD EOI
8376  MAT COPY    R9$()<1,9> TO R1$()<1,9>                           : REM LOAD INTO BUFFR
8380  GOSUB ' 129 (FNI(0),1)                                         : REM STORE ON DISK
8384  INIT(20)    R1$()                                              : REM CLR STORAGE BUFR
8388  INIT(20)    R9$()                                              : REM CLR TEMP BUFR
8392              STR(R9$(1),1,40) = M3$                             : REM LOAD SET NAME
8396              STR(R9$(1),41,1) = HEX(FF)                         : REM LOAD EOI
8400  MAT COPY    R9$()<1,41> TO R1$()<1,41>                         : REM LOAD INTO BUFFR
8404  GOSUB ' 129 (FNI(1),1)                                         : REM STORE ON DISK
8408  RETURN                                                         : REM EXIT
```

```
8412 REM **
8416 REM **
8420  REM **  Subroutine  ********************************************** 2 2 3
8424  REM **  Function >
8428  REM **   To allow the entry of the Symbol.
8432  REM **
8436  REM ***********************************************
8440  DEFFN ' 223                                          : REM ENTRY
8444 REM **
8448 REM **                 ! ALPHANUMERIC !   GRAPHIC   !
8452 REM **                 !--------------!------------!
8456  ON VAL(M4$)-48 GOTO       8468      ,    8524      : REM BRANCH
8460 REM **                 !--------------!------------!
8464 REM **
8468  GOSUB ' 101 (1,1,1,0)                              : REM POSITION CUSUR
8472  PRINT       "Enter Symbol > "                      : REM PRINT PROMPT
8476  GOSUB ' 104 (15,"A","symbol")                      : REM INPUT SYMBOL
8480  GOSUB ' 101 (15,39,1,0)                            : REM POSITION CUSUR
8484  PRINT       "Symbol > ";R2$                        : REM PRINT SYMBOL
8488  INIT(20)    R1$()                                  : REM CLR STORAGE BUFR
8492  INIT(20)    R9$()                                  : REM CLR TEMP BUFR
8496              STR(R9$(1),1,15) = R2$                 : REM LOAD SYMBOL
8500              STR(R9$(1),16,1) = HEX(FF)             : REM LOAD EOI
8504  MAT COPY    R9$()<1,16> TO R1$()<1,16>             : REM LOAD INTO BUFR
8508  GOSUB ' 129 (FN1(2),1)                             : REM STORE ON DISK
8512              N2$ = "DEFFINITION"                    : REM SET NEXT COMMAND
8516  GOTO    8568                                       : REM BRANCH TO EXIT
8520 REM **
8524  GOSUB ' 101 (1,1,1,0)                              : REM POSITION CUSOR
8528  PRINT       "Enter Graphic Symbol > "              : REM PRINT PROMPT
8532  GOSUB ' 101 (3,10,0,0)                             : REM POSITION CUSOR
8536  PRINT       "Graphics Option Unavaiable"           : REM PRINT MSG
8540  GOSUB ' 101 (4,10,0,0)                             : REM POSITION CUSOR
8544  PRINT       "In This Version of AUTHOR"            : REM PRINT MSG
8548  $GIO TONE   /005 (R8$(4),R8$(9))                  : REM PAUSE TONE
8552  $GIO TONE   /005 (R8$(4),R8$(9))                  : REM PAUSE TONE
8556  GOSUB ' 101 (3,10,2,0)                             : REM ERASE MESSAGE
8560  GOTO    8468                                       : REM BRNCH TO ALPHA
8564 REM **
8568  RETURN                                             : REM EXIT
```

```
8572 REM **
8576 REM **
8580  REM **  Subroutine  ********************************************** 2 2 4
8584  REM **  Function >
8588  REM **    To allow the entry of the Definition.
8592  REM ********************************************
8596  DEFFN ' 224                                                    : REM ENTRY
8600  GOSUB ' 101 (14,1,0,0)                                         : REM POSITION CUSOR
8604  INIT(20)    R9$()                                              : REM SET TEMP BUFR '-'
8608              STR(R9$(1),2,19)=" ENTER DEFFINITION"              : REM STOR PROMPT
8612  $GIO PRINT  /005 (R8$(2),R8$(9)) R9$()<1,63>                   : REM PRINT PROMPT
8616  GOSUB ' 106 (FN1(3),1)                                         : REM ENTER & SAVE TEXT
8620  GOSUB ' 101 (14,1,0,0)                                         : REM POSITION CUSOR
8624  INIT(20)    R9$()                                              : REM SET TEMP BUFR '-'
8628  $GIO PRINT  /005 (R8$(2),R8$(9)) R9$()<1,63>                   : REM PRINT PROMPT LINE
8632              N2$ = "MNEMONIC"                                   : REM SET NEXT COMMAND
8636  RETURN                                                         : REM EXIT
```

```
8640 REM **
8644 REM **
8648  REM **  Subroutine  ********************************************** 2 2 5
8652  REM **  Function >
8656  REM **    To allow the entry of the Mnemonic.
8660  REM ***********************************************
8664  DEFFN ' 225                                          : REM ENTRY
8668  GOSUB ' 101 (14,1,0,0)                               : REM POSITION CUSOR
8672  INIT(20)    R9$()                                    : REM SET TEMP BUFR '-'
8676              STR(R9$(1),2,19)=" ENTER MNEMONIC   ":  REM STOR PROMPT
8680  $GIO PRINT  /005 (R8$(2),R8$(9)) R9$()<1,63>         : REM PRINT PROMPT
8684 REM **
8688 REM **               ! ALPHANUMERIC !   GRAPHIC    !
8692 REM **               !--------------!-------------!
8696  ON VAL(M4$)-48 GOTO        8708     ,   8736       : REM BRANCH
8700 REM **               !--------------!-------------!
8704 REM **
8708  GOSUB ' 106 (FNI(4),1)                               : REM ENTER & SAVE TEXT
8712  GOSUB ' 101 (14,1,0,0)                               : REM POSITION CUSOR
8716  INIT(20)    R9$()                                    : REM SET TEMP BUFR '-'
8720  $GIO PRINT  /005 (R8$(2),R8$(9)) R9$()<1,63>         : REM PRINT PROMPT LINE
8724              N2$ = "SIMILAR SYMBOLS"                  : REM SET NEXT COMMAND
8728  GOTO    8772                                         : REM BRANCH TO EXIT
8732 REM **
8736  GOSUB ' 101 (3,10,1,0)                               : REM POSITION CUSOR
8740  PRINT       "Graphics Option Unavaiable"             : REM PRINT MSG
8744  GOSUB ' 101 (4,10,0,0)                               : REM POSITION CUSOR
8748  PRINT       "In This Version of AUTHOR"              : REM PRINT MSG
8752  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
8756  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
8760  GOSUB ' 101 (3,10,2,0)                               : REM ERASE MESSAGE
8764  GOTO    8708                                         : REM ENTR ALPHA TYPE
8768 REM **
8772  RETURN                                               : REM EXIT
```

```
8776 REM **
8780 REM **
8784  REM **  Subroutine  ********************************************** 2 2 6
8788  REM **  Function >
8792  REM **    To allow the entry of similar symbols.
8796  REM ***********************************************
8800  DEFFN ' 226                                          : REM ENTRY
8804  GOSUB ' 101 (13,1,0,0)                               : REM POSITION CUSOR
8808  PRINT   "Touch 'ESCAPE' to End Entry of Similar Symbols."
8812  GOSUB ' 101 (1,1,12,0)                               : REM POSITION CUSOR
8816  PRINT   "Enter Similar Symbols > "                   : REM PRINT PROMPT
8820              K1    = 1                                : REM INIT ROW INDX
8824              K2    = 30                               : REM INIT COL INDX
8828              K7    = 1                                : REM INIT BUFR INDX
8832              K6    = 1                                : REM INIT ITEM INDX
8836  GOSUB ' 101 (K1,K2,1,0)                              : REM POSITION CUSOR
8840  PRINT       K6;"- ";                                 : REM PRINT PROMPT
8844  GOSUB ' 104 (4,"N","9999")                           : REM INPUT SBL NUMBR
8848  IF          R2  = 9999                   THEN 8904   : REM END OF DATA
8852  GOSUB ' 101 (K1,K2,0,0)                              : REM PROCESS INPUT
8856  CONVERT     R2 TO R2$, (####)                        : REM CONV TO DISPLY
8860  PRINT       K6;"- ";R2$
8864              STR(R9$(1),1,3) = M1$                    : REM LOAD SET NUMBR
8868              STR(R9$(1),4,4) = R2$                    : REM LOAD SBL NUMBR
8872  MAT COPY    R9$()<1,7> TO R1$()<K7,7>                : REM LOAD INTO BUFR
8876              K7 = K7 + 7                              : REM INCR BUFR INDX
8880              K1 = K1 + 1                              : REM INCR ROW INDX
8884              K6 = K6 + 1                              : REM INCR ITEM INDX
8888  IF          K1 <> 13                     THEN 8896   : REM NOT BOTOM OF PAG
8892              K1 =  1                                  : REM SET ROW INDX
8896  GOTO        8836                                     : REM INPUT NXT SBL
8900 REM **
8904              STR(R9$(1),1,3) = M1$                    : REM LOAD SET NUMBR
8908              STR(R9$(1),4,4) = R2$                    : REM LOAD EOI FLAG
8912              STR(R9$(1),8,1) = HEX(FF)                : REM LOAD EOI CHAR
8916  MAT COPY    R9$()<1,8> TO R1$()<K7,2016-K7+1>        : REM LOAD BUFFR
8920  GOSUB ' 129 (FNI(5),1)                               : REM SAVE ON DISK
8924              N2$ = "NEXT"                             : REM SET NEXT COMMAND
8928  RETURN                                               : REM EXIT
```

```
8932 REM **
8936 REM **
8940  REM **  Subroutine  ********************************************** 2 2 7
8944  REM **  Function >
8948  REM **    To Increment symbol counter.
8952  REM **********************************************
8956  DEFFN ' 227                                          : REM ENTRY
8960              L2    = L2 + 1                         : REM INCR SBL # INDX
8964  CONVERT     L2 TO M2$, (####)                      : REM CONV TO DISPLY
8968              L1    = 0                              : REM RESET INDEX
8972              L2    = 0                              : REM RESET INDEX
8976              N2$   = "SETNUMBER"                    : REM SET NEXT COMMAND
8980  INIT(20)    Q4$()                                  : REM CLR ITEM FLG BUFR
8984              Q4$(9) = HEX(2A)                       : REM SET ITEM FLG-NEXT
8988  RETURN                                             : REM EXIT
```

```
8992 REM **
8996 REM **
9000 REM **  Subroutine  ********************************************** 2 2 8
9004 REM **  Abstract >
9008 REM **    To delete all items for currently selected
9012 REM **    symbol id.
9016 REM **  Variables >
9020 REM **    L1 > symbol set # indx    ! L2 > symbol numbr indx
9024 REM **
9028 REM ***********************************************
9032  DEFFN ' 228                                          : REM ENTRY
9036  IF          L1*L2 <> 0                THEN 9052      : REM ID'S ARE SET
9040  GOSUB ' 218                                          : REM SET SBL SET #
9044  GOSUB ' 219                                          : REM SET SBL NUMBR
9048  REM **
9052  GOSUB ' 101 (15,15,2,0)                              : REM POSITION CUSOR
9056  PRINTUSING  9108, "> "                               : REM PRINT HEADING
9060  FOR         L9 = 0 TO 5                              : REM BEGIN DEL LOOP
9064   GOSUB '101 (15,50+(L9*2),0,0)                       : REM POSITION CUSOR
9068   PRINTUSING 9112, L9+1                               : REM PRINT FLAGS
9072   $GIO TONE  /005 (R8$(7),R8$(9))                     : REM SOUND TONE
9076   GOSUB '129 (FN1(L9),3)                              : REM DELETE ITEM
9080  NEXT        L9                                       : REM CONT DEL LOOP
9084  REM **
9088              N2$ = "SETNUMBER"                        : REM SET NEXT COMMAND
9092  INIT(20)    G4$()                                    : REM CLR ITEM FLG BUFR
9096  REM **                                                 OUTPUT IMAGE FORMATS
9100  REM **
9104  REM **
9108  %           DELETED SYMBOL ITEM ##
9112  %##
9116  REM **
9120  RETURN                                               : REM EXIT
```

```
9124 REM **
9128 REM **
9132 REM **  Subroutine  ********************************************** 2 3 0
9136 REM **  Abstract >
9140 REM **    To display menu for symbol description input
9144 REM **  Variables >
9148 REM **    Q4$() > item memory flags    ! Q5$() > item key buffer
9152 REM **    R0    > return item value    ! R2$() > srch return bufr
9156 REM **    R2$   > input return value   !
9160 REM **
9164 REM ***********************************************
9168  DEFFN ' 230 (R0)                                        : REM ENTRY
9172  GOSUB ' 101 (1,1,13,0)                                  : REM CLR CRT SCREEN
9176  SELECT      PRINT 005                                   : REM SEL CONSOLE DEV
9180  PRINTUSING  9308, " "                                   : REM PRINT MENU
9184  PRINTUSING  9312, " "
9188  PRINTUSING  9316, Q4$(1), Q4$(6)
9192  PRINTUSING  9320, Q4$(2), Q4$(7)
9196  PRINTUSING  9324, Q4$(3), Q4$(8)
9200  PRINTUSING  9328, Q4$(4), Q4$(9)
9204  PRINTUSING  9332, Q4$(5), Q4$(10)
9208  PRINTUSING  9336,         " "
9212  SELECT      PRINT 405                                   : REM SEL CONSOLE DEV
9216  REM **
9220  GOSUB ' 101 (10,1,0,0)                                  : REM POSITION CUSOR
9224  PRINT       "Enter Command > "                          : REM PRINT INSTRUCTN
9228  GOSUB ' 104 (15,"A",N2$)                                : REM INPUT INSTRUCTN
9232              R0  = LEN(R2$)                              : REM CMPUT CMD LEN
9236  IF          R0 <= 6                        THEN 9244    : REM WITHEN RANGE
9240              R0  = 6                                     : REM SET TO MAX LEN
9244  AND         (STR(R2$,1,R0),DF)                          : REM MASK OUT LWR CASE
9248  MAT SEARCH  Q5$(),=STR(R2$,1,R0) TO R2$() STEP 6:       REM SRCH ID TABLE
9252  IF          R2$(2) <> HEX(0000)          THEN 9260      : REM INVALID COMMAND
9256  IF          R2$(1) <> HEX(0000)          THEN 9284      : REM KEY IS FOUND
9260  GOSUB ' 101 (10,38,0,0)                                 : REM POSITION CUSOR
9264  PRINT       "RE-ENTER"                                  : REM PRINT ERR MSG
9268  $CID TONE   /005 (R8$(4),R8$(9))                        : REM PAUSE TONE
9272  GOSUB ' 101 (10,38,1,0)                                 : REM ERASE ERR MSG
9276  GOTO        9220                                        : REM RE-ENTER CMMND
9280  REM **
9284              R0 = INT(FNR(VAL(R2$(1)))/6) + 1            : REM CMPUT ITEM INDX
9288              Q4$(R0) = HEX(2A)                           : REM SET MENU FLAG
9292  GOSUB ' 101 (1,1,12,0)                                  : REM ERASE MENU
9296  REM **
9300  REM **                                                    OUTPUT IMAGE FORMATS
9304  REM **
9308  %  #                 Command  Summary
9312  %  #  ---------------------------------------------
9316  %       #  SETNUMBER                 #  DEFINITION
9320  %       #  SBLNUMBER                 #  MNEMONIC
9324  %       #  SETNAME                   #  SIMILAR SYMBOLS
9328  %       #  REFERENCE NUMBER          #  NEXT
9332  %       #  SYMBOL                    #  DELETE
9336  %                                    #  EXIT
9340  REM **
9344  RETURN                                                  : REM EXIT
```

8080 - 8072
8216 - 8208
8288 - 8280
8308 - 8304
8468 - 8456 8560
8524 - 8456
8568 - 8516
8708 - 8696 8764
8736 - 8696
8772 - 8728
8836 - 8896
8896 - 8888
8904 - 8848
9052 - 9036
9108 - 9056
9112 - 9068
9220 - 9276
9244 - 9236
9260 - 9252
9284 - 9256
9308 - 9180
9312 - 9184
9316 - 9188
9320 - 9192
9324 - 9196
9328 - 9200
9332 - 9204
9336 - 9208

AUTHOR System: Phase 2 - ATH2LOG2 - V MAP

```
K1     - 8820 8836 8852 8880 8888 8892
K2     - 8824 8836 8852
K6     - 8832 8840 8860 8884
K7     - 8828 8872 8876 8916
L1     - 8056 8968 9036
L2     - 8140 8960 8964 8972 9036
L9     - 9060 9064 9068 9076 9080
M1$    - 8052 8072 8080 8212 8304 8364 8864 8904
M2$    - 8076 8136 8144 8152 8304 8368 8964
M3$    - 8204 8216 8224 8392
M4$    - 8276 8288 8296 8304 8360 8456 8696
N2$    - 8084 8156 8228 8312 8512 8632 8724 8924 8976 9088 9228
Q4$(   - 8088 8980 8984 9092 9188 9192 9196 9200 9204 9288
Q5$(   - 9248
R0     - 9168 9232 9236 9240 9244 9248 9284 9288
R1$(   - 8352 8376 8384 8400 8488 8504 8872 8916
R2     - 8056 8060 8140 8144 8280 8284 8288 8848 8856
R2$    - 8060 8068 8072 8080 8208 8212 8216 8484 8496 8856 8860 8868 8908 9232
         9244 9248
R2$(   - 9248 9252 9256 9284
R8$(   - 8040 8548 8552 8612 8628 8680 8720 8752 8756 9072 9268
R9$(   - 8036 8040 8356 8360 8364 8368 8372 8376 8388 8392 8396 8400 8492 8496
         8500 8504 8604 8608 8612 8624 8628 8672 8676 8680 8716 8720 8864 8868
         8872 8904 8908 8912 8916
```

```
???? DEFFN' 101
       - 8032 8044 8064 8128 8148 8196 8220 8268 8292 8300 8468 8480 8524 8532
         8540 8556 8600 8620 8668 8712 8736 8744 8760 8804 8812 8836 8852 9052
         9064 9172 9220 9260 9272 9292
???? DEFFN' 104
       - 8052 8136 8204 8276 8476 8844 9228
???? DEFFN' 106
       - 8616 8708
???? DEFFN' 129
       - 8380 8404 8508 8920 9076
8024 DEFFN' 218
       - 9040
8124 DEFFN' 219
       - 9044
8192 DEFFN' 220
8264 DEFFN' 221
8348 DEFFN' 222
8440 DEFFN' 223
8596 DEFFN' 224
8664 DEFFN' 225
8800 DEFFN' 226
8956 DEFFN' 227
9032 DEFFN' 228
9168 DEFFN' 230
```

4.5 ATH2LIST

The purpose of the 'ATH2LIST' program is to initialize the symbol table so that the listing program may perform a lookup of the symbol information. This program will also reposition the common table pointer to allow unnecessary variables to be cleared.

The remainder of this section presents the 'ATH2LIST' program listing and reference maps.

AUTHOR System: Phase 2 - ATH2LIST - CNTRL

```
1500  %## Program > AUTHOR/3   (ATH2LIST)   Version > 2.0 2200 T/VP   780321
1504  REM **  Abstract  ***************************************************
1508  REM **                                                               **
1512  REM **    Overlay ATH2LIST will begin listing procecures             **
1516  REM **    by sorting the symbol table into memory.                   **
1520  REM **                                                               **
1524  REM *******************************************************************
1528  DEFFN ' 200                                          : REM ENTRY
1532  GOSUB ' 201                                          : REM INITIALIZE
1536  GOSUB ' 202                                          : REM LOAD SBL TBL
1540  GOSUB ' 203                                          : REM SORT SBL TBL
1544  REM                                                  : REM LOAD NXT LST PROC
1548  COM CLEAR   G4$()                                    : REM CLR COMMON POINTR
1552  LOAD DC T   #1, "ATH2LSTV" 100,300
1556  RETURN                                               : REM EXIT
```

```
1560  REM **
1564  REM **
1568  REM **  Subroutine  ********************************************** 2 0 1
1572  REM **  Abstract >
1576  REM **    To define and initialize sort tables.
1580  REM **  Variables >
1584  REM **    M$()  > internal merge buffer
1588  REM **    C1$() > symbol set memory table
1592  REM **    C2$() > symbol number memory table
1596  REM **    S$()  > sector sort buffer
1600  REM **    L$()  > locator buffer
1604  REM **    V1$() > merge vector 1
1608  REM **    V2$() > merge vector 2
1612  REM **    W$()  > sort work buffer
1616  REM **    Q9$() > general i/o buffer
1620  REM **    Q3$() > symbol table
1624  REM **    L(L) > linear to subscript conversion
1628  REM **    Q0() > symbol count stack
1632  REM **
1636  REM ********************************************
1640  DEFFN ' 201                                          : REM ENTRY
1644  DIM           M$(10,64)4,S$(64)4,W$(64)2             : REM DEFINE BUFRS
1648  DIM           L$(160)2,V1$(11)1,V2$(10)2,B1$(20)1    : REM DEFINE BUFRS
1652  DIM           C1$(10)1,C2$(10)1,T$(1,64)4,B1$1       : REM DEFINE BUFRS
1656  DEFFN L(L)=   INT(VAL(STR(L$(L),2,1))/4)+1           : REM FUNCTION L(L)
1660  DEFFN M(M)=   256*M + VAL(STR(M$(R8,R9),2,1))        : REM FUNCTION M(M)
1664  INIT          (20)M$()                               : REM INIT MRG BUFR
1668  INIT          (00)C1$()                              : REM INIT SET CNTR
1672  INIT          (00)C2$()                              : REM INIT SBL CNTR
1676  INIT          (FF)S$()                               : REM INIT SRT BUFR
1680  INIT          (00)L$()                               : REM INIT LOCATOR
1684  INIT          (01)V1$()                              : REM INIT VECTOR 1
1688  INIT          (00)V2$()                              : REM INIT VECTOR 2
1692  INIT          (00)Q3$()                              : REM INIT SBL TBL
1696  GOSUB ' 101   (5,10,3,0)                             : REM POSITION CUSOR
1700  PRINT         "INITIALIZING SYMBOL TABLE ..."        : REM PRNT TITLE
1704  RETURN                                               : REM EXIT
```

```
1708  REM **
1712  REM **
1716  REM **  Subroutine ********************************************** 2 0 2
1720  REM **  Abstract >
1724  REM **    To Load the Merge buffer from the data base
1728  REM **    index. Each row is sorted.
1732  REM **  Variables >
1736  REM **    Q9$() > i/o buffer     ! S$() > sort buffer
1740  REM **    M$()  > merge buffer   ! L$() > locator buffer
1744  REM **    R1 > indx sctr adrs    ! R2 > subsctr indx
1748  REM **    R9 > scrtch ards rgstr! R8$ > search id
1752  REM **
1756  REM ******************************************************
1760  DEFFN ' 202                                                : REM ENTRY
1764                R8$ = HEX(03F2)                              : REM SET SRCH ID
1768  GOSUB ' 101   (8,10,0,0)                                   : REM POSITION CUSOR
1772  PRINT         "Load Internal Merge Buffer "                : REM PRNT TITLE
1776  GOSUB ' 101   (8,40,0,0)                                   : REM POSITION CUSOR
1780  PRINT         "Sector     Sub-Sector"                      : REM PRNT COL HEADS
1784  FOR           R1 = 1 TO 10                                 : REM BEG INDX LOOP
1788  GOSUB ' 101   (9,42,0,0)                                   : REM POSITION CUSOR
1792  PRINT         R1                                           : REM PRNT SCTR ADRS
1796  GOSUB ' 101   (9,55,0,0)                                   : REM POSITION CUSOR
1800  DATA LOAD     BA T #2, (R1,R9)T$()                         : REM LOAD INDX SCTR
1804  MAT SEARCH    T$(), >= R8$ TO L$() STEP 4                  : REM SRCH I/O BUFR
1808  FOR           R2 = 1 TO 65                                 : REM BEG SUBINDX LOOP
1812  IF            L$(R2) <> HEX(0000)        THEN 1832         : REM END OF BUFR
1816  IF            R2 <> 1                    THEN 1824         : REM NOT BEG OF ROW
1820                V1$(R1) = HEX(FF)                            : REM SET TO EMPTY ROW
1824                R2 = 65                                      : REM SET TO END
1828  GOTO          1840                                         : REM BRNCH TO NXT LOOP
1832                BIN(STR(L$(R2),1,1)) = 1                     : REM CONV TO SUBSCRPT
1836                BIN(STR(L$(R2),2,1)) = FNL(R2)               : REM CONV TO SUBSCRPT
1840  CONVERT       R2 TO R4$, (##)                              : REM CONV TO DSPLY
1844  PRINT         R4$;HEX(0808)                                : REM PRNT SBSCTR INDX
1848  NEXT          R2                                           : REM CONT SUBINDX LOOP
1852  MAT MOVE      T$(), L$(1)   TO S$(1)                       : REM LOAD SORT BUFR
1856  MAT SORT      S$()          TO W$(), L$()                  : REM SORT BUFR
1860  MAT MOVE      S$() , L$(1)  TO M$(R1,1)                    : REM LOAD MERGE BUFR
1864  INIT          (FF)S$()                                     : REM INIT SORT BUFR
1868  INIT          (00)L$()                                     : REM INIT LOCATOR BUFR
1872  NEXT          R1                                           : REM CONT INDX LOOP
1876  RETURN                                                     : REM EXIT
```

```
1880  REM **
1884  REM **
1888  REM **  Subroutine  ********************************************* 2 0 3
1892  REM **  Abstract >
1896  REM **    To Sort the merge buffer creating the sorted
1900  REM **    symbol table.
1904  REM **  Variables >
1908  REM **    M$() > merge buffer  ! V1$() > work vectr 1
1912  REM **    V2$() > work vectr 2 ! L$()  > locator buffer
1916  REM **    R1 > symbol tbl indx ! R2 > symbol tbl indx
1920  REM **    R3 > bufr out indx   ! R5 > elements merged
1924  REM **    R4 > merge end flag  ! R6 > sbl id tbl indx
1928  REM **
1932  REM *********************************************
1936  DEFFN ' 203                                          : REM ENTRY
1940              R1 = 1                                   : REM SBL TBL ROW INDX
1944              R2 = 1                                   : REM SBL TBL COL INDX
1948              R3 = 1                                   : REM BUFR OUT INDX
1952              R6 = 0                                   : REM SBL ID INDX
1956  GOSUB ' 101 (8,10,2,0)                               : REM POSITION CUSOR
1960  PRINT      "Load Symbol Table From Merge Buffer":     REM PRNT TITLE
1964  GOSUB ' 101 (9,20,0,0)                               : REM POSITION CUSOR
1968  PRINT       "Symbol >"                               : REM PRNT TITLE
1972  GOSUB ' 101 (10,30,0,0)                              : REM POSITION CUSOR
1976  PRINT       "Set  Number  Item"                      : REM PRNT COL HEADS
1980  GOSUB ' 101 (11,30,0,0)                              : REM POSITION CUSOR
1984  MAT MERGE   M$() TO V1$(), V2$(), L$()               : REM BEG MERGE
1988  IF          L$(1) =  HEX(0000)          THEN 2088    : REM ELEMENTS MERGED
1992              R4 = VAL(V1$(11))                        : REM FETCH MRG FLAG
1996  IF          R4 = 0                      THEN 2004    : REM LOCATOR FULL
2000              V1$(R4) = HEX(FF)                        : REM MRG ROW EMPTY
2004              R5 = 160                                 : REM ELEMENTS TO MOV
2008  MAT MOVE    M$()(3,2), L$(1), R5 TO Q3$(R1,R2)       : REM LOAD SBL TBL
2012              R0$ = HEX(FFFF)                          : REM SRCH CHARACTER
2016  MAT SEARCH  Q3$()<R3,900-R3+1>,=R0$TO R2$()STEP 2:   REM SRCH FOR END
2020  IF          R2$(1) <> HEX(0000)         THEN 2036    : REM UNEVEN LOAD
2024  IF          R3 > 900                    THEN 2088    : REM BUFFER FULL
2028              R0$ = HEX(0000)                          : REM SET SRCH STRING
2032  GOTO        2016                                     : REM BRANCH TO SRCH
2036              R5 = INT((FNR(VAL(R2$(1)))-1)/2)         : REM CMPUT END POS
2040              R3 = R3 + R5*2                           : REM INCR BUFOUT INDX
2044  GOSUB ' 206                                          : REM INCR SBL CNTRS
2048  IF          R3 > 899                    THEN 2088    : REM SORT COMPLETE
2052              R1 = INT((R3-1)/12) + 1                  : REM INCR ROW INDX
2056              R2 = INT((R3 - (R1-1)*12)/2) + 1         : REM INCR COL INDX
2060              Q0(1) = VAL(C1$(1))                      : REM STOR # OF SETS
2064              Q0(2) = VAL(C2$(1))                      : REM STOR # OF SBLS
2068  GOSUB ' 101 (13,10,0,0)                              : REM POSITION CUSOR
2072  PRINT       "Symbols Sets > ";Q0(1);"      ";        : REM PRNT # OF SETS
2076  PRINT       "      Symbols > ";Q0(2)                 : REM PRNT # OF SBLS
2080  GOSUB ' 101 (11,30,0,0)                              : REM POSITION CUSOR
2084  GOTO        1984                                     : REM CONT MERGE
2088  RETURN                                               : REM EXIT
```

```
2092  REM **
2096  REM **
2100  REM **  Subroutine  ********************************************** 2 0 6
2104  REM **  Abstract >
2108  REM **    To count the number of symbol sets and number
2112  REM **    of symbols.
2116  REM **  Variables >
2120  REM **    R5 > # of items merged ! R0 > sub-item number
2124  REM **    R8 > set number        ! R9 > symbl number
2128  REM **    R7 > packed ref #      ! R2$ > scratch bufr
2132  REM **    B1$ > temp bit patern  ! C1$() > cnt buf sets
2136  REM **    C2$() > cnt buf sbls   ! B1$() > intrnl mem tbl
2140  REM **    L$() > locator bufr    ! M$() > merge buffer
2144  REM **    R6   > sbl id tbl indx ! G7$()> sbl id tbl
2148  REM **
2152  REM ********************************************
2156  DEFFN ' 206                                          : REM ENTRY
2160  IF          R5 = 0                        THEN 2272  : REM MRG COMPLETED
2164  FOR         R4 = 1 TO R5                             : REM BEG COUNT LOOP
2168              R8 = VAL(STR(L$(R4),1,1))                : REM CMPUT ROW SBSCRPT
2172              R9 = VAL(STR(L$(R4),2,1))                : REM CMPUT COL SBSCRPT
2176              R7 = FNM(VAL(M$(R8,R9)))                 : REM FETCH REF #
2180              R8 = INT(R7/1000)                        : REM CMPUT SET #
2184              R9 = INT((R7-(R8*1000))/10)              : REM CMPUT SBL #
2188              R0 = R7 - (R8*1000 + R9*10)              : REM CMPUT ITM #
2192  INIT        (20)R2$                                  : REM INIT DSPLY BUFR
2196  CONVERT     R8 TO STR(R2$,1,2), (##)                 : REM CONV TO DSPLY
2200  CONVERT     R9 TO STR(R2$,8,2), (##)                 : REM CONV TO DSPLY
2204  CONVERT     R0 TO STR(R2$,15,2), (##)                : REM CONV TO DSPLY
2208  PRINT       STR(R2$,1,16)                            : REM PRNT DSPLY BUFR
2212  INIT        (08)R2$                                  : REM INIT DSPLY BUFR
2216  PRINT       STR(R2$,1,16)                            : REM PRNT DSPLY BUFR
2220  MAT COPY    C1$() TO B1$()                           : REM STOR MEM TBL
2224  GOSUB ' 207 (B1$,B1$(1),R8)                          : REM CHECK MEM TBL
2228  IF          B1$ <> HEX(00)                THEN 2244  : REM SET COUNTED
2232  MAT COPY    B1$() TO C1$()                           : REM STOR MEM TBL
2236              BIN(C1$(1)) = VAL(C1$(1)) + 1            : REM INCR COUNT
2240  $TRAN       (C2$()<2,9>, R4$()<181,4>)00 R           : REM ZERO SBL CNT
2244  MAT COPY    C2$() TO B1$()                           : REM STOR MEM TBL
2248  GOSUB ' 207 (B1$,B1$(1),R9)                          : REM CHECK MEM TBL
2252  IF          B1$ <> HEX(00)                THEN 2268  : REM SET COUNTED
2256  MAT COPY    B1$() TO C2$()                           : REM STOR MEM TBL
2260              BIN(C2$(1)) = VAL(C2$(1)) + 1            : REM INCR COUNT
2264              R6 = R6 + 1                              : REM INCR ID INDX
2268  NEXT        R4                                       : REM CONT COUNTING
2272  RETURN                                               : REM EXIT
```

```
2276  REM **
2280  REM **
2284  REM **  Subroutine  ********************************************** 2 0 7
2288  REM **  Abstract >
2292  REM **   To set a bit on corsponding to the item number.
2296  REM **   If bit is on return a zero bit patern.
2300  REM **  Variables >
2304  REM **    B1$ > temp storage byte ! B2 > item number
2308  REM **    B1$() > memory table    ! B0 > scratch indx
2312  REM **    B4 > bit position       !
2316  REM **
2320  REM ***********************************************
2324  DEFFN ' 207   (B1$,B1$(1),B2)                          : REM ENTRY
2328              B0 = INT((B2-1)/8) + 2                     : REM CMPUT ELEMNT
2332              B4 = B2 - (B0-2)*8                         : REM CMPUT BIT POS
2336  GOSUB ' 208   (B1$(B0),B4)                             : REM MASK BYTE
2340  IF            B1$ <> HEX(00)              THEN 2356    : REM BIT WAS ON
2344  GOSUB ' 209   (B1$(B0),B4)                             : REM TURN BIT ON
2348              B1$(B0) = B1$                              : REM STOR NEW BIT PTRN
2352              B1$ = HEX(00)                              : REM RESET FLAG
2356  RETURN                                                 : REM EXIT
```

```
2360  REM **
2364  REM **
2368  REM **  Subroutine  ********************************************** 2 0 8
2372  REM **  Abstract >
2376  REM **    To mask a given bit position.
2380  REM **  Variables >
2384  REM **    B1$ > storage byte    ! B4 > bit position
2388  REM **
2392  REM ***********************************************
2396  DEFFN ' 208  (B1$,B4)                                  : REM ENTRY
2400  REM **
2404  REM **       ! BIT 80 ! BIT 40 ! BIT 20 ! BIT 10 !
2408  REM **       !--------!--------!--------!--------!
2412  ON B4 GOTO     2428  ,  2432  ,  2436  ,  2440  ,
                     2444  ,  2448  ,  2452  ,  2456  : REM BRANCH
2416  REM **       !--------!--------!--------!--------!
2420  REM **       ! BIT 04 ! BIT 04 ! BIT 02 ! BIT 01 !
2424  REM **
2428  AND          (B1$,80)                 : GOTO 2460 : REM MASK BIT 8
2432  AND          (B1$,40)                 : GOTO 2460 : REM MASK BIT 7
2436  AND          (B1$,20)                 : GOTO 2460 : REM MASK BIT 6
2440  AND          (B1$,10)                 : GOTO 2460 : REM MASK BIT 5
2444  AND          (B1$,08)                 : GOTO 2460 : REM MASK BIT 4
2448  AND          (B1$,04)                 : GOTO 2460 : REM MASK BIT 3
2452  AND          (B1$,02)                 : GOTO 2460 : REM MASK BIT 2
2456  AND          (B1$,01)                 : GOTO 2460 : REM MASK BIT 1
2460  RETURN                                             : REM EXIT
```

```
2464  REM **
2468  REM **
2472  REM **  Subroutine  ********************************************** 2 0 9
2476  REM **  Abstract >
2480  REM **    To turn on a given bit position.
2484  REM **  Variables >
2488  REM **    B1$ > storage byte    ! B4 > bit position
2492  REM **
2496  REM **********************************************
2500  DEFFN ' 209  (B1$,B4)                                  : REM ENTRY
2504  REM **
2508  REM **       ! BIT 80 ! BIT 40 ! BIT 20 ! BIT 10 !
2512  REM **       !--------!--------!--------!--------!
2516  ON B4 GOTO     2532  ,  2536  ,  2540  ,  2544  ,
                     2548  ,  2552  ,  2556  ,  2560  : REM BRANCH
2520  REM **       !--------!--------!--------!--------!
2524  REM **       ! BIT 04 ! BIT 04 ! BIT 02 ! BIT 01 !
2528  REM **
2532  OR           (B1$,80)                : GOTO 2564 : REM MASK BIT 8
2536  OR           (B1$,40)                : GOTO 2564 : REM MASK BIT 7
2540  OR           (B1$,20)                : GOTO 2564 : REM MASK BIT 6
2544  OR           (B1$,10)                : GOTO 2564 : REM MASK BIT 5
2548  OR           (B1$,08)                : GOTO 2564 : REM MASK BIT 4
2552  OR           (B1$,04)                : GOTO 2564 : REM MASK BIT 3
2556  OR           (B1$,02)                : GOTO 2564 : REM MASK BIT 2
2560  OR           (B1$,01)                : GOTO 2564 : REM MASK BIT 1
2564  RETURN                                           : REM EXIT
```

AUTHOR System: Phase 2 - ATH2LIST - # MAP

```
(0100) - 1552
(0300) - 1552
 1824  - 1816
 1832  - 1812
 1840  - 1828
 1984  - 2084
 2004  - 1996
 2016  - 2032
 2036  - 2020
 2088  - 1988 2024 2048
 2244  - 2228
 2268  - 2252
 2272  - 2160
 2356  - 2340
 2428  - 2412
 2432  - 2412
 2436  - 2412
 2440  - 2412
 2444  - 2412
 2448  - 2412
 2452  - 2412
 2456  - 2412
 2460  - 2428 2432 2436 2440 2444 2448 2452 2456
 2532  - 2516
 2536  - 2516
 2540  - 2516
 2544  - 2516
 2548  - 2516
 2552  - 2516
 2556  - 2516
 2560  - 2516
 2564  - 2532 2536 2540 2544 2548 2552 2556 2560
```

AUTHOR System: Phase 2 - ATH2LIST - V MAP

```
B0     - 2328 2332 2336 2344 2348
B1$    - 1652 2224 2228 2248 2252 2324 2340 2348 2352 2396 2428 2432 2436 2440
         2444 2448 2452 2456 2500 2532 2536 2540 2544 2548 2552 2556 2560
B1$(   - 1648 2220 2224 2232 2244 2248 2256 2324 2336 2344 2348
B2     - 2324 2328 2332
B4     - 2332 2336 2344 2396 2412 2500 2516
C1$(   - 1652 1668 2060 2220 2232 2236
C2$(   - 1652 1672 2064 2240 2244 2256 2260
L      - 1656
L$(    - 1648 1656 1680 1804 1812 1832 1836 1852 1856 1860 1868 1984 1988 2008
         2168 2172
M      - 1660
M$(    - 1644 1660 1664 1860 1984 2008 2176
Q0(    - 2060 2064 2072 2076
Q3$(   - 1692 2008 2016
Q4$(   - 1548
R0     - 2188 2204
R0$    - 2012 2016 2028
R1     - 1784 1792 1800 1820 1860 1872 1940 2008 2052 2056
R2     - 1808 1812 1816 1824 1832 1836 1840 1848 1944 2008 2056
R2$    - 2192 2196 2200 2204 2208 2212 2216
R2$(   - 2016 2020 2036
R3     - 1948 2016 2024 2040 2048 2052 2056
R4     - 1992 1996 2000 2164 2168 2172 2268
R4$    - 1840 1844
R4$(   - 2240
R5     - 2004 2008 2036 2040 2160 2164
R6     - 1952 2264
R7     - 2176 2180 2184 2188
R8     - 1660 2168 2176 2180 2184 2188 2196 2224
R8$    - 1764 1804
R9     - 1660 1800 2172 2176 2184 2188 2200 2248
S$(    - 1644 1676 1852 1856 1860 1864
T$(    - 1652 1800 1804 1852
V1$(   - 1648 1684 1820 1984 1992 2000
V2$(   - 1648 1688 1984
W$(    - 1644 1856
```

AUTHOR System: Phase 2 - ATH2LIST - ' MAP

```
???? DEFFN' 101
       - 1696 1768 1776 1788 1796 1956 1964 1972 1980 2068 2080
1528 DEFFN' 200
1640 DEFFN' 201
       - 1532
1760 DEFFN' 202
       - 1536
1936 DEFFN' 203
       - 1540
2156 DEFFN' 206
       - 2044
2324 DEFFN' 207
       - 2224 2248
2396 DEFFN' 208
       - 2336
2500 DEFFN' 209
       - 2344
```

4.6 ATH2LSTV

The purpose of the 'ATH2LSTV' program is to define a new common table containing the necessary variables needed by the listing program.

The remainder of this section presents the 'ATH2LSTV' program listing.

```
100  %## Program > AUTHOR/2  (ATH2LSTV)  Version > 2.0 2200 T/VP   780321
104  REM ** Abstract ************************************************************
108  REM **                                                                      **
112  REM **    This routine will clear program text and                          **
116  REM **    load new common definitions.                                      **
120  REM **                                                                      **
124  REM ************************************************************************
128  REM **
132  REM ********************************************** PROGRAM STORAGE
136  REM **
140  REM **
144  COM          D0, D1   , D2, D3  , D4, D5, D6 , D7, D8, D9
148  COM          E1       , E1$4    , E2$4  , N1$, N2$
152  COM          H8$(23)45, K5$(1)64, S4$(1)64
156  COM          R0, R1   , R2, R3  , R4, R5, R6 , R7, R8, R9
160  COM          R0$2     , R1$2    , R2$64      , R3$8
164  COM          R4$64    , R5$1    , Q1$(18)2   , Q5$(12)6
168  COM          R6$64    , R7$10   , R8$2       , R9$64
172  COM          R0$(1)1  , R2$(2)2 , R4$(20)16  , R5$(10)1
176  COM          R6$(16)1 , R7$(64)1, R8$(9)24   , R9$(1)64
180  COM          S0$2     , S1$2    , S2$2       , S3$2
184  COM          S4$2     , S5$2    , S6$2       , S7$2
188  COM          Q0(2)    , Q0$(20)1, Q3$(75,6)2
192  COM          Q4$(11)1 , Q7$(5)2
196  COM          Q8$(128)2, Q9$(64)4
200  COM          W2$, W3$ , W4$, W5$, W6$
204  COM          R1$(4,64)1,R3$(1)64, Q2$(25,75)1
208  REM **
212  REM **
216  %##          END OF COMMON STORAGE
220  REM **
224  REM **
300  LOAD DC T  #1, "ATH2LSTP" 1500,9999
```

4.7 ATH2LSTP

The purpose of the 'ATH2LSTP' program is to provide the control logic and listing routines necessary for outputting the data base to the line printer. The 'ATH2LSTP' program is presented in six parts.

4.7.1 ATH2LST1

This section of the 'ATH2LSTP' program contains the main branching instructions for the execution of the listing commands. This section also contains subroutines 100 to 123.

The remainder of this section presents the 'ATH2LST1' program listing and reference maps.

AUTHOR System: Phase 2 - ATH2LST1 - CNTRL

```
1500  %##  Program > AUTHOR/2  (ATH2LST1)   Version > 2.0 2200 T/VP  780321
1504  REM **  Abstract  ************************************************************
1508  REM **                                                                       **
1512  REM **    This routine will provide a copy of the Phase 2                    **
1516  REM **    data base.  One or all of the data base items                      **
1520  REM **    may be listed. The avaiable list commands are >                    **
1524  REM **       TITLE        ORGANIZATION       TYPE             SYMBOL       **
1528  REM **       SUBTITLE     TRAINING           NAME             HELP         **
1532  REM **       AUTHORS      IMPORTANCE         SOURCE           LIST         **
1536  REM **       DATE         USAGE              EXAMPLE          ESCAPE       **
1540  REM **                                                                       **
1544  REM ************************************************************************
1548  REM **
1552  GOSUB ' 200                                                : REM PRINT HEADR PAGE
1556  GOSUB ' 105 (R0,"Data Listing Procedures")                 : REM INPUT COMMAND
1560  PRINT       HEX(03)                                        : REM CLR CRT SCREEN
1564  GOSUB ' 201                                                : REM TEST PRINTR ON
1568  GOSUB ' 101 (14,1,0,0)                                     : REM POSITION CUSOR
1572 REM **
1576 REM **          ! TITLE  ! SUBTITL ! AUTHOR ! DATE ! ORGAN !
1580 REM **          !--------!---------!--------!------!-------!
1584  ON R0-0  GOTO   1632   , 1636    , 1640   , 1644 , 1648  ,
                      1652   , 1656    , 1660   , 1664 , 1668  : REM BRANCH
1588 REM **          !--------!---------!--------!------!-------!
1592 REM **          ! TRAIN  ! IMPORTN ! USAGE  ! TYPE ! NAME  !
1596 REM **
1600 REM **
1604 REM **          ! SOURCE ! EXAMPLE ! SYMBOL ! HELP ! LIST  !
1608 REM **          !--------!---------!--------!------!-------!
1612  ON R0-10 GOTO 1672   , 1676    , 1680   , 1684 , 1688  ,
                      1692   , 1696    , 1696   , 1696 , 1696  : REM BRANCH
1616 REM **          !--------!---------!--------!------!-------!
1620 REM **          ! ESCAPE ! NULL    ! NULL   ! NULL ! NULL  !
1624 REM **
```

```
1628  REM
1632  GOSUB ' 134 (1,001,4,"subtitle")      : GOTO 1556 : REM TITLE
1636  GOSUB ' 134 (1,002,5,"author")        : GOTO 1556 : REM SUBTITLE
1640  GOSUB ' 134 (1,003,6,"date")          : GOTO 1556 : REM AUTHORS
1644  GOSUB ' 134 (1,004,7,"organization") : GOTO 1556 : REM DATE
1648  GOSUB ' 134 (1,005,8,"training")      : GOTO 1556 : REM ORGANIZATION
1652  GOSUB ' 134 (1,011,9,"importance")    : GOTO 1556 : REM TRAINING
1656  GOSUB ' 134 (1,031,10,"usage")        : GOTO 1556 : REM IMPORTANCE
1660  GOSUB ' 134 (1,032,11,"type")         : GOTO 1556 : REM USAGE
1664  GOSUB ' 134 (1,022,13,"name")         : GOTO 1556 : REM TYPE
1668  GOSUB ' 134 (1,021,12,"source")       : GOTO 1556 : REM NAME
1672  GOSUB ' 134 (1,023,14,"example")      : GOTO 1556 : REM SOURCE
1676  GOSUB ' 134 (1,041,15,"symbol")       : GOTO 1556 : REM EXAMPLE
1680  GOSUB ' 202                           : GOTO 1556 : REM SYMBOL
1684  GOSUB ' 203                           : GOTO 1556 : REM HELP
1688  GOSUB ' 204                           : GOTO 1708 : REM LIST
1692  GOSUB ' 205                           : GOTO 1708 : REM ESCAPE
1696  GOSUB ' 203                           : GOTO 1556 : REM NULL CMND
1700  REM **
1704  REM **
1708  PRINT       HEX(03)                                 : REM CLR CRT SCREEN
1712  PRINT       "A U T H O R    S Y S T E M";HEX(0D0A)
1716  PRINT       "P h a s e      II";HEX(0D0A0A0A)
1720  PRINT       "* * * *  LISTING COMPLETED  * * * *"
1724  PRINT       HEX(010C0C0C0C)
1728  STOP                                                : REM END OF LIST
1732  END                                                 : REM END OF PROGRAM
```

```
1736  REM **
1740  REM **
1744  REM ************************************************************************
1748  REM **              L A B E L E D     S U B R O U T I N E S                 **
1752  REM ************************************************************************
1756 REM **
1760 REM **
1764 REM **  Subroutine  ********************************************** 1 0 6
1768 REM **  Abstract >
1772 REM **    This routine will load the printer page buffer
1776 REM **    with text. The text will be loaded according to the
1780 REM **    row and colum location specified.
1784 REM **  Variables >
1788 REM **    S1 > row location indx      ! S2 > col location indx
1792 REM **    S3 > text length  indx      ! S4$() > text buffer
1796 REM **    S6 > linear scale indx      ! Q2$() > printer pag bufr
1800 REM **
1804 REM ********************************************
1808  DEFFN ' 106 (S1,S2,S3,S4$(1))                          : REM ENTRY
1812  IF          S3 <> 0                    THEN 1820       : REM STR LEN NOT ZERO
1816              S3 =  LEN(S4$(1))                          : REM CMPUT STR LENGTH
1820              S6 =  (S1-1)*75 + S2                       : REM CMPUT LOCATION
1824  MAT COPY    S4$() <1,S3> TO Q2$() <S6,S3>              : REM TRANSFER TO BUFR
1828  RETURN                                                 : REM EXIT
```

```
1832 REM **
1836 REM **
1840 REM **  Subroutine  ********************************************** 1 0 7
1844 REM **  Abstract >
1848 REM **    This routine will load storage buffer with data
1852 REM **    item from disk.
1856 REM **  Variables >
1860 REM **    E1 > begin load indx          ! I1 > data item id
1864 REM **    I4 > return item flage        ! RO$ > begining dsk location
1868 REM **    RO > byte size of dat item    !
1872 REM **
1876 REM ***********************************************
1880  DEFFN ' 107 (E1,I1)                                    : REM ENTRY
1884  IF          E1 <> 1                        THEN 1896 : REM CONT BUFR LOAD
1888  GOSUB ' 130 (I1)                                     : REM PERFORM INDX LOKUP
1892  IF          I4 =  0                        THEN 1904 : REM ITEM NOT FOUND
1896  IF          E1 =  0                        THEN 1904 : REM ITEM LOADED
1900  GOSUB ' 131 (RO$,RO,I1,E1)                           : REM LOAD DATA ITEM
1904  RETURN                                               : REM EXIT
```

```
1908 REM **
1912 REM **
1916 REM **  Subroutine  ********************************************** 1 0 8
1920 REM **  Abstract >
1924 REM **      To Output page buffer onto the selected
1928 REM **      output device.
1932 REM **      Output index code       1 - CRT   N/A
1936 REM **                              2 - TTY   N/A
1940 REM **                              3 - LPT
1944 REM **                              4 - END OF LPT FILE
1948 REM **  Variables  >
1952 REM **      W6 > output device index     ! RO$() > input register
1956 REM **      W7 > undrscr lokup indx      ! W2$   > return scan flag
1960 REM **      W8 > bufer out lin indx      ! W3$   > return scan flag
1964 REM **                                   ! W4$   > return scan flag
1968 REM **
1972 REM ************************************************
1976  DEFFN ' 108 (W6)                                    : REM ENTRY
1980  GOSUB ' 124                                         : REM SEL SPL CHAR SCAN
1984 REM **
1988 REM **        !   CRT   !   TTY   !   LPT   !   END   !
1992 REM **        !---------!---------!---------!---------!
1996  ON W6 GOTO     2072  ,  2072  ,  2016  ,  2056       : REM BRANCH
2000 REM **        !---------!---------!---------!---------!
2004 REM **
2008 REM **                                                 OUTPUT TO LPT DEVICE
2012 REM **
2016  SELECT #5   015                                      : REM SEL OUTPUT DEV
2020  SELECT      TAPE 07B                                  : REM SEL TAPE DEV
2024  GOSUB ' 118                                         : REM SEL UNDRSCR MODE
2028  GOSUB ' 119 (1)                                     : REM PRINT HEADER
2032  FOR         W8 = 1 TO 25                            : REM BEGIN BUFR LOOP
2036  GOSUB ' 117                                         : REM PRINT TEXT LINE
2040  NEXT        W8                                      : REM INCR LINE INDX
2044  GOSUB ' 119 (2)                                     : REM PRINT PAGE NUMBR
2048  IF          W4$ = "NO UNDERSCORE"     THEN 2060     : REM NO UNDERSCORING
2052  IF          W4$ = "OVERLAY"           THEN 2060     : REM IF OVRLAY OUTPUT
2056  GOSUB ' 120                                         : REM PRINT OVERLAY
2060  GOTO        2072                                    : REM BRANCH TO EXIT
2064 REM **
2068 REM **
2072  RETURN                                              : REM EXIT
```

```
2076 REM **
2080 REM **
2084 REM **  Subroutine  ********************************************** 1 0 9
2088 REM **  Abstract >
2092 REM **    To scan text buffer and create lookup table and
2096 REM **    overlay buffer for selected output devices.
2100 REM **  Variables >
2104 REM **    Q2$() > output text buffer     !
2108 REM **    W3$() > underscore lookup tbl ! W4$() > conversion buffer
2112 REM **    X4 > end of loop indx (row)    ! X5 > end of loop indx (col)
2116 REM **    R4 > lookup tbl indx (row)     ! R5 > lookup tbl indx (col)
2120 REM **    R6 > linear buff position      ! R5$() > single char storage
2124 REM **    W6 > output device indx        ! W9 > buffer size
2128 REM **
2132 REM ********************************************
2136  DEFFN ' 109                                          : REM ENTRY
2140  DIM          W3$(1,200)2,W9$(7)8                     : REM DEFINE WORK BUFRS
2144  DIM          X3$(1,075)2,Y3$(1,075)2                 : REM DEFINE WORK BUFRS
2148               W9$(1) = HEX(5FA0)                      : REM INIT CONV VALUES
2152               W9$(2) = HEX(20800900)                  : REM INIT CONV VALUES
2156               W9$(3) = HEX(20802D812F825F83)          : REM INIT CONV VALUES
2160               W9$(4) = HEX(2000200120022003)          : REM INIT CONV VALUES
2164               W9$(5) = HEX(2020)                      : REM INIT CONV VALUES
2168               W9 = 185                                : REM SET SIZE TO MAX
2172               R2$(1) = "Z"                            : REM DEFINE SPL CHAR 1
2176               R2$(2) = "O"                            : REM DEFINE SPL CHAR 2
2180  $TRAN        (Q2$()<1,W9>, W9$()<1,8>) R             : REM MINIZE UNDRSCRS
2184  MAT SEARCH   Q2$()<1,W9>, > R5$(3) TO W3$()          : REM CREAT LOKUP TBL 1
2188  IF           W6 = 1                    THEN 2196     : REM NOT TTY OUTPUT
2192  $TRAN        (Q2$()<1,W9>, W9$()<1,8>) 7F R          : REM CLR HIGH BIT
2196  MAT SEARCH   Q2$()<1,W9>, = R2$(1) TO X3$()          : REM CREAT LOKUP TBL 2
2200  MAT SEARCH   Q2$()<1,W9>, = R2$(2) TO Y3$()          : REM CREAT LOKUP TBL 3
2204               W3$(1,200)   = HEX(0000)                : REM SET EOI BYTE
2208               X3$(1,075)   = HEX(0000)                : REM SET EOI BYTE
2212               Y3$(1,075)   = HEX(0000)                : REM SET EOI BYTE
2216  IF           W6$ = "NULL"              THEN 2296     : REM NULL CHAR SCAN
2220  IF           W6$ = "BOTH"              THEN 2232     : REM EXECUTE SCAN
2224  IF           W6$ <> "ZEE"              THEN 2264     : REM NULL 'Z' SCAN
2228  REM **
2232  FOR          R4 = 1 TO 1                             : REM BEGIN TBL SCAN ROW
2236  FOR          R5 = 1 TO 075                           : REM BEGIN TBL SCAN COL
2240  GOSUB ' 125 (R4,R5,1,075,X3$(R4,R5),HEX(81))         : REM SCAN LOKUP TABLE 1
2244  NEXT         R5                                      : REM CONT  TBL SCAN COL
2248  NEXT         R4                                      : REM CONT  TBL SCAN ROW
2252               X3$(X6,X7) = HEX(FFFF)                  : REM SET EOI BYTE
2256  IF           W6$ = "BOTH"              THEN 2268     : REM EXECUTE SCAN
2260  REM **
2264  IF           W6$ <> "ZERO"             THEN 2296     : REM NULL 'O' SCAN
2268  FOR          R4 = 1 TO 1                             : REM BEGIN TBL SCAN ROW
2272  FOR          R5 = 1 TO 075                           : REM BEGIN TBL SCAN COL
2276  GOSUB ' 125 (R4,R5,1,075,Y3$(R4,R5),HEX(82))         : REM SCAN LOKUP TABLE 2
2280  NEXT         R5                                      : REM CONT  TBL SCAN COL
2284  NEXT         R4                                      : REM CONT  TBL SCAN ROW
2288               Y3$(X6,X7) = HEX(FFFF)                  : REM SET EOI BYTE
```

```
2292  REM **
2296  FOR          R4 = 1 TO 1                                  : REM BEGIN TBL SCAN ROW
2300  FOR          R5 = 1 TO 200                                : REM BEGIN TBL SCAN COL
2304  GOSUB ' 125 (R4,R5,1,200,W3$(R4,R5),HEX(83))              : REM SCAN LOKUP TABLE 0
2308  NEXT         R5                                           : REM CONT  TBL SCAN COL
2312  NEXT         R4                                           : REM CONT  TBL SCAN ROW
2316               W3$(X6,X7) = HEX(FFFF)                       : REM SET EOI BYTE
2320 REM **
2324 REM **       !   CRT   !   TTY   !   LPT   !
2328 REM **       !---------!---------!---------!
2332  ON W6 GOTO     2356   ,  2356   ,  2344                   : REM BRANCH
2336 REM **       !---------!---------!---------!
2340 REM **
2344  $TRAN        (Q2$()<1,W9>, W9$()<17,24>) 83 R             : REM CONV OVRLAY BUFR
2348  GOTO         2356                                         : REM BRANCH TO EXIT
2352 REM **
2356  RETURN                                                    : REM EXIT
```

```
2360 REM **
2364 REM **
2368 REM **  Subroutine  ********************************************** 1 1 7
2372 REM **  Abstract >
2376 REM **    To print a line without trailing blanks on selected
2380 REM **    output device.
2384 REM **  Variables >
2388 REM **    W9 > temp text bufr indx  ! Q2$() > text buffer
2392 REM **    R9$() > temp line bufr    ! R5$() > single char storage
2396 REM **    R2$() > serach retrn bufr ! W6 > output device indx
2400 REM **    WO > temp length of line  ! W7 > null line flag
2404 REM **
2408 REM ********************************************
2412  DEFFN ' 117                                          : REM ENTRY
2416  DIM          W9$(7)8,S9$(75)1                        : REM DEFINE WORK BUFR
2420               W9$(1) = HEX(5FA020202020)              : REM DEFINE CONVERSN
2424               W9 = (W8-1)*75 + 1                      : REM CMPUT BEG OF LIN
2428  $TRAN        (Q2$()<W9,75>, W9$()<1,8>) R            : REM MINIZE UNDERSCOR
2432  MAT COPY     -Q2$()<W9,75> TO S9$()<1,75>            : REM REVERSE COPY
2436  MAT SEARCH   S9$(), <> R5$(1) TO R2$()               : REM SCAN FOR NONBLANKS
2440  IF           R2$(1) <> HEX(0000)       THEN 2452     : REM TRAILING BLANKS
2444               BIN(STR(R2$(1),1,1)) = 0                : REM SET TO NULL LINE
2448               BIN(STR(R2$(1),2,1)) = 76               : REM SET TO NULL LINE
2452               WO = 75 - (FNR(VAL(R2$(1)))-1)          : REM CMPUT LEN OF STR
2456               BIN(STR(R8$(9),6,1)) = WO               : REM STOR LEN OF STR
2460               BIN(STR(R8$(9),7,1)) = 00               : REM SET LIMIT
2464  $GIO SETUP   #5 (R8$(8),R8$(9))                      : REM SETUP CR OR LF
2468 REM **
2472 REM **        !   CRT   !   TTY   !   LPT   !
2476 REM **        !---------!---------!---------!
2480  ON W6 GOTO     2512   ,  2512   ,  2492              : REM BRANCH
2484 REM **        !---------!---------!---------!
2488 REM **
2492  INIT(20)     R9$()                                   : REM CLR TAB BUFFER
2496  $GIO TAB     #5 (14671608A000,R8$(9))R9$()<1,0>      : REM HORZ TAB
2500  $GIO TEXT    #5 (A00042504000,R8$(9))Q2$()<W9,WO>:   REM PRINT LINE
2504  GOTO         2512                                    : REM BRANCH TO EXIT
2508 REM **
2512  RETURN                                               : REM EXIT
```

```
2516 REM **
2520 REM **
2524 REM **  Subroutine  ********************************************** 1 1 8
2528 REM **  Abstract >
2532 REM **    To Display Instructions to set up LPT device.
2536 REM **  Variables >
2540 REM **    W4$ > return scan flag      ! W4$() > i/o buffer #1
2544 REM **    W9  > scratch loop indx     ! W5$() > i/o buffer #2
2548 REM **    RO$() > input register      !
2552 REM **
2556 REM ***********************************************
2560  DEFFN ' 118                                          : REM ENTRY
2564  DIM          X9$(10)45                               : REM DEFINE WORK BUFRS
2568  IF           W4$ = "NO UNDERSCORE"      THEN 2696    : REM SKIP ROUTINE
2572  IF           W4$ = "UNDERSCORE"         THEN 2696    : REM SKIP ROUTINE
2576  IF           W4$ = "OVERLAY"            THEN 2696    : REM SKIP ROUTINE
2580  IF           W4$ = "TAPE"               THEN 2696    : REM SKIP ROUTINE
2584  IF           W4$ = "SET PRINTER"        THEN 2644    : REM INIT PRINTER ONLY
2588               W4$ = "NO UNDERSCORE"                   : REM SET RETRN SCAN FLG
2592  PRINT        HEX(03)                                 : REM CLR CRT SCREEN
2596  GOSUB ' 101 (5,1,0,0)                                : REM POSITION CUSOR
2600  SELECT       PRINT 005                               : REM SEL OUTPUT MODE
2604  PRINTUSING   2676, " "                               : REM PRINT INSTRUTN 1
2608  PRINTUSING   2680, " "                               : REM PRINT INSTRUTN 2
2612  PRINTUSING   2684, " "                               : REM PRINT INSTRUTN 3
2616  SELECT       PRINT 405                               : REM SEL OUTPUT MODE
2620  GOSUB ' 101 (2,1,0,0)                                : REM POSITION CUSOR
2624  PRINTUSING   2688, " ";                              : REM PRINT PROMPT
2628  KEYIN        RO$(1), 2632, 2632      : GOTO 2628     : REM RESPONCE WAIT
2632  IF           RO$(1) <> "1"              THEN 2640    : REM SCAN FLAG SET
2636               W4$ = "UNDERSCORE"                      : REM SET SCAN FLAG
2640  IF           RO$(1) <> "2"              THEN 2656    : REM SCAN FLAG SET
2644               W4$ = "TAPE"                            : REM SET SCAN FLAG
2648  REM **
2652  GOSUB ' 127                                          : REM TEST TAPE READY
2656  GOSUB ' 128                                          : REM TEST PRINTER READY
2660  GOSUB ' 129                                          : REM PRINT INSTRUCTIONS
2664 REM **
2668 REM **                                                  INSTRUCITON FORMATS
2672 REM **
2676  % LPT UNDERSCORE MENU #  1 -  SINGLE PAGE UNDERSCORING
2680  %                        #  2 -  MULTI PAGE UNDERSCORING (TAPE)
2684  %                        #  3 -  DO NOT SCAN FOR UNDERSCORING
2688  % # Enter Code from LPT Underscoring Menu >
2692  REM **
2696  RETURN                                               : REM EXIT
```

```
2700 REM **
2704 REM **
2708 REM **  Subroutine  ********************************************** 1 1 9
2712 REM **  Abstrct >
2716 REM **    To print upper/lower hash lines on the LPT device.
2720 REM **  Variables >
2724 REM **    W7$() > hash contorl buffer  ! W9 > upper/lower index
2728 REM **
2732 REM ***********************************************
2736  DEFFN ' 119 (W9)                                          : REM ENTRY
2740  DIM          W7$(19)4                                     : REM DEFINE WORK BUFRS
2744  INIT(20)     W7$()                                        : REM CLR TABLE
2748       W7$(1)=HEX(30303030) : W7$(2)=HEX(0A0A0A0A)          : REM INIT TABLE
2752 REM **
2756 REM **        !   Header   ! Page Num. !
2760 REM **        !------------!------------!
2764  ON W9 GOTO       2776     ,   2808                       : REM BRANCH
2768 REM **        !------------!------------!
2772 REM **
2776  $GIO TOP     #5 (400CA000     ,R8$(9))W7$()<5,2>  : REM TOP OF FORM
2780  $GIO TAB     #5 (A000         ,R8$(9))W7$()<9,15> : REM HORZ TAB
2784  $GIO HEAD    #5 (A000400D400A,R8$(9))H8$()<1,45> : REM PRINT HEADING
2788  $GIO TAB     #5 (A000         ,R8$(9))W7$()<9,15> : REM HORZ TAB
2792  $GIO SUBHEAD #5 (A000400D400A,R8$(9))H8$()<46,45>: REM PRINT SUBHEADING
2796  $GIO VTAB    #5 (A000         ,R8$(9))W7$()<5,2>  : REM VERTICAL TAB
2800  GOTO         2836                                 : REM BRANCH TO EXIT
2804 REM **
2808  IF           W4$ = "OVERLAY"            THEN 2836 : REM NULL HEADER
2812  CONVERT      PO TO W7$(1), (####)                 : REM CREATE PAG NUMBR
2816  $GIO VTAB    #5 (A000         ,R8$(9))W7$()<5,1>  : REM VERTICAL TAB
2820  $GIO HTAB    #5 (A000         ,R8$(9))W7$()<9,38> : REM HORZ TAB
2824  $GIO PAGNUM  #5 (A000400D     ,R8$(9))W7$()<1,4>  : REM PRINT PAGE NUMBR
2828               PO = PO + 1                          : REM INCR PAGE INDEX
2832 REM **
2836  RETURN                                            : REM EXIT
```

```
2840 REM **
2844 REM **
2848 REM **  Subroutine  ********************************************** 1 2 0
2852 REM **  Abstract >
2856 REM **    To print the underscore overlay buffer on to LPT device.
2860 REM **  Variables >
2864 REM **    W6 > output device indx    ! W4$ > return scan flag
2868 REM **    Z3 > end of file flag      !
2872 REM **
2876 REM ***********************************************
2880  DEFFN ' 120                                          : REM ENTRY
2884  IF          W6 = 4                       THEN 2936 : REM END OF PRNT FILE
2888  GOSUB ' 109                                        : REM CREATE OVERLAY
2892  IF          W4$ = "TAPE"                 THEN 2924 : REM WRITE ON TAPE
2896 REM **
2900  GOSUB ' 122                                        : REM DEVICE WAIT
2904              W4$ = "OVERLAY"                        : REM SET SCAN FLAG
2908  GOSUB ' 108 (W6)                                   : REM PRINT BUFFR
2912              W4$ = "UNDERSCORE"                     : REM RESET SCAN FLAG
2916  GOTO        2984                                   : REM BRANCH TO EXIT
2920 REM **
2924  GOSUB ' 190 (12,1,Z3,1875,1,6)                     : REM WRITE BUFFER
2928  GOTO        2984                                   : REM BRANCH TO EXIT
2932 REM **
2936  GOSUB ' 190 (10,3,Z3,0,1,6)                        : REM WRITE EOF
2940  DEFFN ' 016                                        : REM RECOVERY ENTRY
2944  GOSUB ' 190 (9,3,Z3,0,1,6)                         : REM REWIND TAPE
2948              W6 = 3                                 : REM RESET DEVICE INDX
2952  GOSUB ' 122                                        : REM DEVICE WAIT
2956              W4$ = "OVERLAY"                        : REM SET SCAN FLAG
2960  GOSUB ' 190 (7,1,Z3,1875,1,6)                      : REM READ RECORD
2964  IF          Z3 = 4                       THEN 2976 : REM END OF FILE
2968  GOSUB ' 108 (W6)                                   : REM PRINT BUFFER
2972  GOTO        2960                                   : REM READ NXT RECORD
2976  GOSUB ' 190 (9,2,Z3,0,1,6)                         : REM REWIND TAPE
2980 REM **
2984  RETURN                                             : REM EXIT
```

```
2988 REM **
2992 REM **
2996 REM **  Subroutine  ********************************************** 1 2 2
3000 REM **  Abstract >
3004 REM **    To allow the output device to be reset.  Program will
3008 REM **    pause until the device is disabled then will
3012 REM **    continue until a device ready is sensed.
3016 REM **  Variables >
3020 REM **    W9$() > micro command buffer  ! W6 > output device indx
3024 REM **
3028 REM ***********************************************
3032  DEFFN ' 122                                          : REM ENTRY
3036  DIM          W9$(7)8                                 : REM DEFINE WORK BUFR
3040               W9$(1) = HEX(122141002020)              : REM INIT MICRO CMND
3044               W9$(2) = HEX(122145002020)              : REM INIT MICRO CMND
3048               W9$(3) = HEX(122141002020)              : REM INIT MICRO CMND
3052               W9$(4) = HEX(44657669636520 20)         : REM INIT SCRN CHAR
3056               W9$(5) = HEX(24202020526561 64)         : REM INIT SCRN CHAR
3060               W9$(6) = HEX(79202020202420 20)         : REM INIT SCRN CHAR
3064               W9$(7) = HEX(5761697420202020)          : REM INIT SCRN CHAR
3068  $GIO TONE    /005 (R8$(4),R8$(9))                    : REM ATTENTION TONE
3072  GOSUB ' 123                                          : REM PRNT SCRN CHAR
3076  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
3080  $IF ON       #5,                              3068 : REM IF READY
3084  GOSUB ' 123                                          : REM PRNT SCRN CHAR
3088  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
3092  $IF ON       #5,                              3100 : REM IF READY
3096                                              GOTO 3084 : REM IF NOT READY
3100  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
3104  $IF ON       #5,                              3112 : REM IF READY
3108                                              GOTO 3084 : REM IF NOT READY
3112  RETURN                                               : REM EXIT
```

```
3116 REM **
3120 REM **
3124 REM **  Subroutine  ********************************************** 1 2 3
3128 REM **  Abstract >
3132 REM **    To Display a large cusor on console screen while
3136 REM **    waitting for device ready.
3140 REM **  Variables >
3144 REM **    W9$() > char storage bufr    ! W9 > temp bufr index (loop)
3148 REM **    R9$() > scratch cusor bufr   ! K1 > random row position
3152 REM **    K2 > random col position     !
3156 REM **
3160 REM ***********************************************
3164  DEFFN ' 123                                          : REM ENTRY
3168  INIT(08)     R9$()                                   : REM INIT CUSUR BUFR
3172               K1 = INT(RND(-1)*10)                    : REM GEN RND POS
3176  IF           K1 = 0                        THEN 3172 : REM INVALID POS
3180               K2 = INT(RND(-1)*100)                   : REM GEN RND POS
3184  IF           K2 > 55                       THEN 3180 : REM INVALID POS
3188  $GIO CLR     /005(4003          ,R8$(9))             : REM CLR CRT SCREEN
3192  GOSUB ' 101 (K1,K2,0,0)                              : REM POSITION CUSUR
3196  FOR          W9 = 25 TO 49 STEP 6                    : REM BEG CHAR PRNT
3200  $GIO PRINT   /005(4006A000      ,R8$(9))W9$()<W9,6>: REM PRINT LINE
3204  $GIO BAKSPC /005(4006400AA000,R8$(9))R9$()<1,6>    : REM POSITION FOR NXT
3208  NEXT         W9                                      : REM PRNT NXT LINE
3212  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 2 - ATH2LST1 - # MAP

```
1556  - 1632 1636 1640 1644 1648 1652 1656 1660 1664 1668 1672 1676 1680 1684
        1696
1632  - 1584
1636  - 1584
1640  - 1584
1644  - 1584
1648  - 1584
1652  - 1584
1656  - 1584
1660  - 1584
1664  - 1584
1668  - 1584
1672  - 1612
1676  - 1612
1680  - 1612
1684  - 1612
1688  - 1612
1692  - 1612
1696  - 1612
1708  - 1688 1692
1820  - 1812
1896  - 1884
1904  - 1892 1896
2016  - 1996
2056  - 1996
2060  - 2048 2052
2072  - 1996 2060
2196  - 2188
2232  - 2220
2264  - 2224
2268  - 2256
2296  - 2216 2264
2344  - 2332
2356  - 2332 2348
2452  - 2440
2492  - 2480
2512  - 2480 2504
2628  - 2628
2632  - 2628
2640  - 2632
2644  - 2584
2656  - 2640
2676  - 2604
2680  - 2608
2684  - 2612
2688  - 2624
2696  - 2568 2572 2576 2580
2776  - 2764
2808  - 2764
2836  - 2800 2808
2924  - 2892
2936  - 2884
2960  - 2972
2976  - 2964
2984  - 2916 2928
3068  - 3080
3084  - 3096 3108
3100  - 3092
```

3112 - 3104
3172 - 3176
3180 - 3184

```
2940 DEFFN' 16
???? DEFFN' 101
       - 1568 2596 2620 3192
???? DEFFN' 105
       - 1556
1808 DEFFN' 106
1880 DEFFN' 107
1976 DEFFN' 108
       - 2908 2968
2136 DEFFN' 109
       - 2888
2412 DEFFN' 117
       - 2036
2560 DEFFN' 118
       - 2024
2736 DEFFN' 119
       - 2028 2044
2880 DEFFN' 120
       - 2056
3032 DEFFN' 122
       - 2900 2952
3164 DEFFN' 123
       - 3072 3084
???? DEFFN' 124
       - 1980
???? DEFFN' 125
       - 2240 2276 2304
???? DEFFN' 127
       - 2652
???? DEFFN' 128
       - 2656
???? DEFFN' 129
       - 2660
???? DEFFN' 130
       - 1888
???? DEFFN' 131
       - 1900
???? DEFFN' 134
       - 1632 1636 1640 1644 1648 1652 1656 1660 1664 1668 1672 1676
???? DEFFN' 190
       - 2924 2936 2944 2960 2976
???? DEFFN' 200
       - 1552
???? DEFFN' 201
       - 1564
???? DEFFN' 202
       - 1680
???? DEFFN' 203
       - 1684 1696
???? DEFFN' 204
       - 1688
???? DEFFN' 205
       - 1692
```

4.7.2 ATH2LST2

This section of the 'ATH2LSTP' program contains subroutines 124 to 129.

The remainder of this section presents the 'ATH2LST2' program listing and reference maps.

```
4000 %## Program > AUTHOR/2  (ATH2LST2)  Version > 2.0  2200 T/VP  780321
4003 REM **
4006 REM **  Subroutine  ********************************************** 1 2 4
4009 REM **  Abstract >
4012 REM **    This routine will select the type of special
4015 REM **    character scanning to be executed.
4018 REM **  Variables >
4021 REM **    W6$ > scaning flag bufr ! RO$() > char input bufr
4024 REM **
4027 REM **********************************************
4030  DEFFN ' 124                                          : REM ENTRY
4033  IF          W6$ = "ZERO"              THEN 4159 : REM SCAN SELECTED
4036  IF          W6$ = "ZEE"               THEN 4159 : REM SCAN SELECTED
4039  IF          W6$ = "BOTH"              THEN 4159 : REM SCAN SELECTED
4042  IF          W6$ = "NULL"              THEN 4159 : REM SCAN SELECTED
4045  REM **
4048  PRINT       HEX(03)                              : REM CLR CRT SCREEN
4051  GOSUB ' 101 (5,1,0,0)                            : REM POSITION CUSOR
4054  SELECT      PRINT 005                            : REM SEL DISPLY MODE
4057  PRINTUSING  4141, " "                            : REM PRINT MENU
4060  PRINTUSING  4144, " "
4063  PRINTUSING  4147, " "
4066  PRINTUSING  4150, " "
4069  SELECT      PRINT 405                            : REM SEL DISPLY MODE
4072  GOSUB ' 101 (2,1,0,0)                            : REM POSITION CUSOR
4075  PRINTUSING  4153, " "
4078  KEYIN       RO$(1), 4081, 4081      : GOTO 4078 : REM RESPONCE WAIT
4081  PRINT       HEX(03)                              : REM CLR CRT SCREEN
4084              W6$ = "NULL"                         : REM SET DEFLT VALUE
4087  IF          RO$(1) <> "1"             THEN 4096 : REM SET '0' SCAN
4090              W6$ = "ZERO"
4093  REM **
4096  IF          RO$(1) <> "2"             THEN 4105 : REM SET 'Z' SCAN
4099              W6$ = "ZEE"
4102  REM **
4105  IF          RO$(1) <> "3"             THEN 4114 : REM SET '0-Z' SCAN
4108              W6$ = "BOTH"
4111  REM **
4114  IF          W6$ = "NULL"              THEN 4159 : REM BRNCH TO EXIT
4117  IF          W6 <> 3                   THEN 4159 : REM NOT LPT OUTPUT
4120              W4$ = "SET PRINTER"                  : REM SET SCAN FLAG
4123  SELECT #5   015                                  : REM SEL LINE PRINTER
4126  SELECT TAPE 07B                                  : REM SEL TAPE DRIVE
4129  GOSUB ' 118                                      : REM INIT PRINTER
4132  REM **
4135  REM **                                             OUTPUT IMAGE FORMATS
4138  REM **
4141  % # Special Character Scanning >   1 - Scan for zero (0 <- /)
4144  % #                               2 - Scan for zee  (Z <- -)
4147  % #                               3 - Scan for both '0' & 'Z'
4150  % #                               4 - NO Special Scanning
4153  % # Enter Scanning Code from Below Table >
4156  REM **
4159  RETURN                                           : REM EXIT
```

```
4162 REM **
4165 REM **
4168 REM **  Subroutine  ********************************************** 1 2 5
4171 REM **  Abstract >
4174 REM **    This routine will scan search search return buffer
4177 REM **    and insert special character into text buffer in
4180 REM **    location specified by buffer value.
4183 REM **  Variables >
4186 REM **    R4 > lookup tbl row indx  ! R5 > lookup tbl col indx
4189 REM **    X6 > rtrn row indx on 0   ! X7 > rtrn col indx on 0
4192 REM **    X8 > max table row indx   ! X9 > max table col indx
4195 REM **    R2$() > search bufr value ! RO$() > spcl insert char
4198 REM **    R6 > decimal bufr value   !
4201 REM **
4204 REM ********************************************
4207  DEFFN ' 125 (R4,R5,X8,X9,R2$(1),RO$(1))          : REM ENTRY
4210              R6 = FNR(VAL(R2$(1)))               : REM CMPUT DECMAL POS
4213  IF          R6 <> 0                  THEN 4228  : REM NOT END OF TABLE
4216              X6 = R4    :   X7 = R5              : REM SAVE END OF TABLE
4219              R4 = X8    :   R5 = X9              : REM SEND END OF LOOP
4222  GOTO 4234                                       : REM BRNCH TO EXIT
4225  REM **
4228  IF          W6 = 2                   THEN 4234  : REM TTY OUTPUT
4231  MAT COPY    RO$()<1,1> TO G2$()<R6,1>           : REM INSRT SPCL CHAR
4234  RETURN                                          : REM EXIT
```

```
4237 REM **
4240 REM **
4243 REM **  Subroutine  ********************************************** 1 2 6
4246 REM **  Abstract >
4249 REM **    This routine will print the text string buffer
4252 REM **    X9$() as specified by input string parameters.
4255 REM **  Variables >
4258 REM **    X8 > beging string position ! X9 > length of string
4261 REM **
4264 REM ********************************************
4267  DEFFN ' 126 (X8,X9)                                  : REM ENTRY
4270  $GIO PRINT  #5 (A000    ,R8$(9))X9$()<X8,X9>         : REM PRINT TEXT STRING
4273  RETURN                                               : REM EXIT
```

```
4276 REM **
4279 REM **
4282 REM **  Subroutine  ********************************************** 1 2 7
4285 REM **  Abstract >
4288 REM **    This routine will test the tape unit for ready status.
4291 REM **  Variables >
4294 REM **    W4$ > underscor status flag ! W6$ > special char scan flag
4297 REM **    W6 > output device flag     !
4300 REM **
4303 REM ***********************************************
4306  DEFFN ' 127                                          : REM ENTRY
4309  $GIO TST ON (6CF5440086077105122140O0,R8$(9))       : REM SEND TEST SIGNAL
4312  $IF      ON /07B, 4363                               : REM TAPE READY
4315              W4$ = " "                                : REM CLR SCAN FLAG
4318              W6$ = " "                                : REM CLR SCAN FLAG
4321              W6  = 8                                  : REM SET FOR ERR RECVRY
4324  GOSUB ' 101 (11,10,0,0)                             : REM POSITION CUSOR
4327  PRINTUSING  4354, " "                               : REM PRINT ERR MSG
4330  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
4333  GOSUB ' 101 (13,1,0,0)                              : REM POSITION CUSOR
4336  PRINTUSING  4357, " "                               : REM PRINT ERR MSG
4339  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
4342  GOTO        4312                                    : REM CONT TEST
4345  REM **
4348  REM **                                                OUTPUT IMAGE FORMATS
4351  REM **
4354  % # * * * *   TAPE UNIT NOT READY   * * * *
4357  % # Touch 'RESET'... Then Type   RUN 1500   and Touch 'RETURN'
4360  REM **
4363  RETURN                                               : REM EXIT
```

```
4366 REM **
4369 REM **
4372 REM **  Subroutine  ********************************************** 1 2 8
4375 REM **  Abstract >
4378 REM **    This routine will test the printer for ready status.
4381 REM **  Variables >
4384 REM **    R8$() > i/o micro command buffer !
4387 REM **
4390 REM ***********************************************
4393  DEFFN ' 128                                          : REM ENTRY
4396  $GIO READY  #5 (4100         ,R8$(9))               : REM SEND CTL CHAR
4399  $IF  ON     #5, 4417                                : REM DEVICE READY
4402  GOSUB ' 101 (11,10,0,0)                             : REM POSITION CUSUR
4405  PRINTUSING  4432, " "                               : REM PRINT ERR MSG
4408  $GIO PAUSE  /005(R8$(4),R8$(9))                     : REM ERROR TONE
4411  GOSUB ' 101 (11,10,1,0)                             : REM ERASE ERR MSG
4414  GOTO        4396                                    : REM CONT READY TEST
4417  PRINT       HEX(030A0A0A0A0A0A)                     : REM CLR CONSOLE DEV
4420  PRINTUSING  4435, " "                               : REM PRINT INSTRUCTN
4423  REM **
4426  REM **                                                OUTPUT IMAGE FORMATS
4429  REM **
4432  %          * * * *  LINEPRINTER IS NOT READY  * * * *        #
4435  % * * * *  FOLLOW INSTRUCTIONS ON OUTPUT DEVICE  * * * *    #
4438  REM **
4441  RETURN                                               : REM EXIT
```

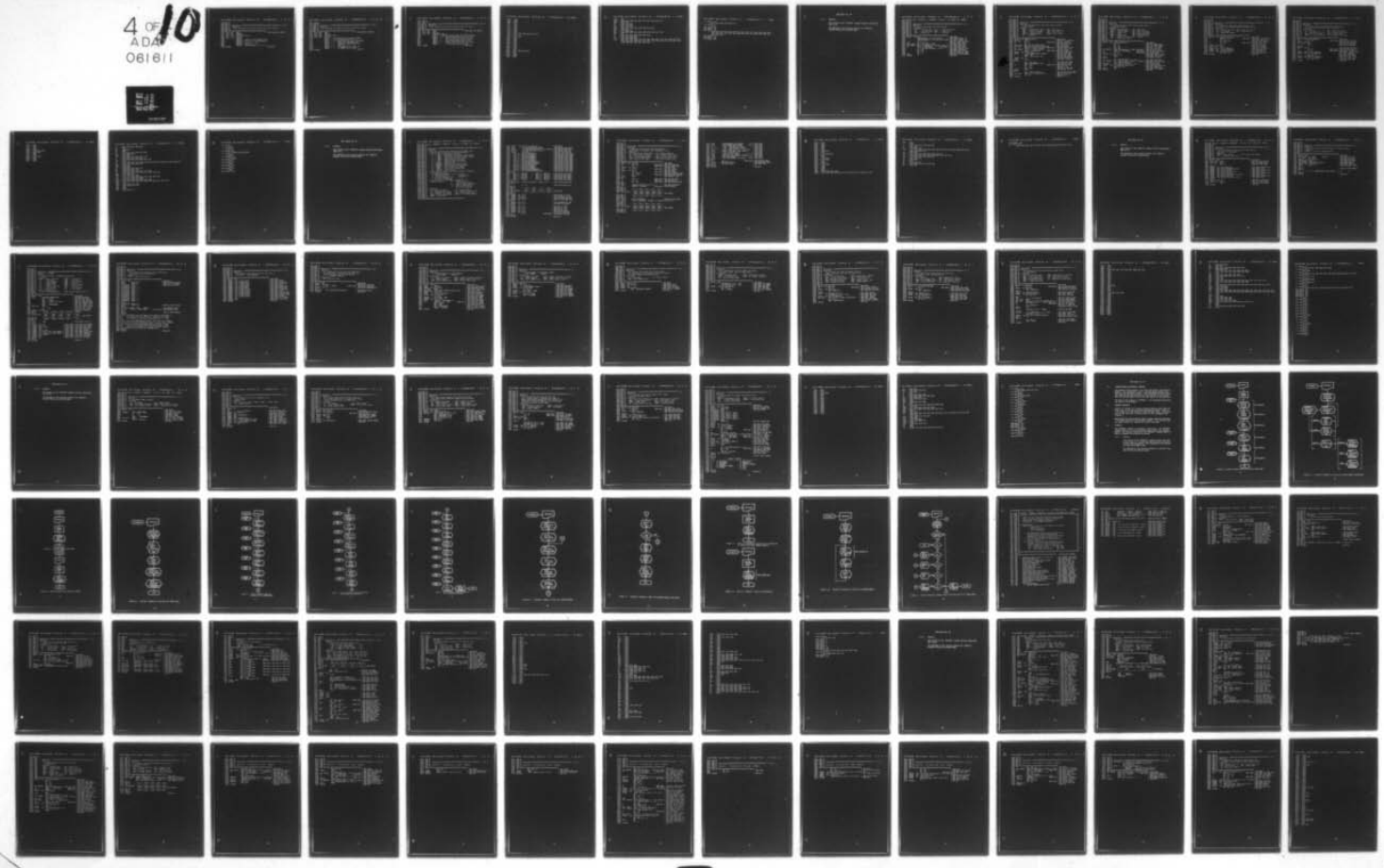
```
4444 REM **
4447 REM **
4450 REM **  Subroutine  ********************************************** 1 2 9
4453 REM **  Abstract >
4456 REM **    This routine will display the alignment instructions
4459 REM **    on the lineprinter.
4462 REM **  Variables >
4465 REM **    X7 > temp text bufr indx   ! X9$() > temp text bufr
4468 REM **    RO$() > char input bufr    !
4471 REM **
4474 REM ************************************************
4477  DEFFN ' 129                                          : REM ENTRY
4480  GOSUB 4672                                           : REM INIT TEXT BUFFR 1
4483  GOSUB ' 126 ( 34,12)                                 : REM VERT TAB
4486  GOSUB ' 126 ( 46,30)                                 : REM LEFT ARROWS
4489  GOSUB ' 126 (361, 5)                                 : REM HORZ TAB
4492  GOSUB ' 126 (136,30)                                 : REM TEXT
4495  GOSUB ' 126 ( 91,30)                                 : REM RIGHT ARROWS
4498  GOSUB ' 126 ( 91,35)                                 : REM RIGHT ARROWS
4501  GOSUB ' 126 ( 44, 2)                                 : REM CRLF
4504  REM **
4507  GOSUB ' 126 (181,30)                                 : REM LEFT MARGIN
4510  GOSUB ' 126 (226,40)                                 : REM SIZE
4513  GOSUB ' 126 (361,35)                                 : REM HORZ TAB
4516  GOSUB ' 126 (316,25)                                 : REM RIGHT MARGIN
4519  GOSUB ' 126 ( 44, 2)                                 : REM CRLF
4522  REM **
4525  GOSUB ' 126 (361,00)                                 : REM HORZ TAB
4528  GOSUB ' 126 (181,30)                                 : REM LEFT MARGIN
4531  GOSUB ' 126 (271,20)                                 : REM SIZE
4534  GOSUB ' 126 (316,25)                                 : REM RIGHT MARGIN
4537  GOSUB ' 126 ( 44, 2)                                 : REM CRLF
4540  REM **
4543  REM **
4546  GOSUB 4726                                           : REM INIT TEXT BUFFR 2
4549  GOSUB ' 126 (285,10)                                 : REM HORZ TAB
4552  GOSUB ' 126 (406,40)                                 : REM HORZ LINE
4555  GOSUB ' 126 (406,40)                                 : REM HORZ LINE
4558  GOSUB ' 126 ( 44, 2)                                 : REM CRLF
4561  REM **
4564  FOR         X7 = 2 TO 9                              : REM BEG LINE LOOP
4567  GOSUB ' 126 (285,30)                                 : REM HORZ TAB
4570  GOSUB ' 126 (((X7-1)*45)+1,LEN(X9$(X7)))             : REM TEXT
4573  GOSUB ' 126 ( 44, 2)                                 : REM CRLF
4576  NEXT        X7                                       : REM CONT LINE LOOP
4579  REM **
4582  GOSUB ' 126 (285,10)                                 : REM HORZ TAB
4585  GOSUB ' 126 (406,40)                                 : REM HORZ LINE
4588  GOSUB ' 126 (406,40)                                 : REM HORZ LINE
4591  GOSUB ' 126 ( 44, 2)                                 : REM CRLF
```

```
4594  REM **
4597  REM **
4600  GOSUB 4786                                          : REM INIT TEXT BUFFR 3
4603  FOR         X7 = 2 TO 8                             : REM BEGIN LINE LOOP
4606  GOSUB ' 126 (361,30)                                : REM HORZ TAB
4609  GOSUB ' 126 (((X7-1)*45)+1,LEN(X9$(X7)))            : REM TEXT
4612  GOSUB ' 126 ( 44, 2)                                : REM CRLF
4615  NEXT        X7                                      : REM CONT LINE LOOP
4618  REM **
4621  REM **
4624  $GIO TOP    #5 (400C    ,R8$(9))                    : REM ISSUE FORM FEED
4627  KEYIN       RO$(1), 4642, 4627                      : REM RESPONCE WAIT
4630  IF          RO$(1) = HEX(81)          THEN 4627     : REM NO TONE
4633  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
4636  GOTO        4627                                    : REM CONTINUE WAIT
4639  REM **
4642  IF          RO$(1) = HEX(84)          THEN 4480     : REM REPRINT PAGE
4645  IF          RO$(1) = HEX(81)          THEN 4627     : REM SHUT OFF TONE
4648  RETURN                                              : REM EXIT
```

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U)
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

4 OF 10
ADA
061611



```
4651 REM **
4654 REM **
4657 REM **  Subroutine  ********************************************** 1 2 9.1
4660 REM **  Abstract >
4663 REM **    This routine will load text buffer with text group 1
4666 REM **
4669 REM ***********************************************
4672  INIT (20)    X9$()                                  : REM LOAD TEXT BUFR 1
4675  INIT (0A)    X9$(01)
4678  INIT (0D)    STR(X9$(1),45,1)
4681  INIT ("<")   X9$(02)
4684  INIT (">")   X9$(03)
4687               X9$(04) = "Align to Sixth Paper Hole"
4690               X9$(05) = "<<<<<------- Left Margin"
4693               X9$(06) = "14 7/8  X  11"
4696               X9$(07) = "08 1/2  X  11"
4699               X9$(08) = "Right Margin ------->>>>>"
4702  RETURN                                              : REM EXIT
```

```
4705  REM **
4708  REM **
4711 REM **  Subroutine  ********************************************** 1 2 9.2
4714 REM **  Abstract >
4717 REM **    This routine will load text buffer with text group 2
4720 REM **
4723 REM **********************************************
4726  INIT (20)    X9$()                                  : REM LOAD TEXT BUFR 2
4729  INIT (0A)    X9$(1)
4732  INIT (0D)    STR(X9$(1),45,1)
4735  INIT (2D)    X9$(10)
4738               X9$(2)  = "This page should now be aligned"
4741               X9$(3)  = "to the desired position..."
4744               X9$(4)  = "--  To Continue to print THIS PAGE"
4747               X9$(5)  = "    TOUCH the 'CONTINUE' Key."
4750               X9$(6)  = "--  To Begin Listing TOUCH the"
4753               X9$(7)  = "    'RUN' Key."
4756               X9$(8)  = "--  This page should appear on"
4759               X9$(9)  = "    an UPWARD Facing Page."
4762  RETURN                                              : REM EXIT
```

```
4765  REM **
4768  REM **
4771 REM **  Subroutine  ********************************************** 1 2 9.3
4774 REM **  Abstract >
4777 REM **    This routine will load text buffer with text group 3
4780 REM **
4783 REM ***********************************************
4786  INIT (20)    X9$()                                  : REM LOAD TEXT BUFR 3
4789  INIT (0A)    X9$(1)
4792  INIT (0D)    STR(X9$(1),45,1)
4795               X9$(2)  = "Special Instructions for Underscoring:"
4798               X9$(3)  = "--  Single Page Overlaying:  At the"
4801               X9$(4)  = "    sound of the tone, position page"
4804               X9$(5)  = "    two inches above top of form."
4807               X9$(6)  = "--  Multi Page Overlaying:  At the"
4810               X9$(7)  = "    sound of the tone, position paper"
4813               X9$(8)  = "    to this page and position."
4816  RETURN                                              : REM EXIT
```

AUTHOR System: Phase 2 - ATH2LST2 - # MAP

4078 - 4078
4081 - 4078
4096 - 4087
4105 - 4096
4114 - 4105
4141 - 4057
4144 - 4060
4147 - 4063
4150 - 4066
4153 - 4075
4159 - 4033 4036 4039 4042 4114 4117
4228 - 4213
4234 - 4222 4228
4312 - 4342
4354 - 4327
4357 - 4336
4363 - 4312
4396 - 4414
4417 - 4399
4432 - 4405
4435 - 4420
4480 - 4642
4627 - 4627 4630 4636 4645
4642 - 4627
4672 - 4480
4726 - 4546
4786 - 4600

```
Q2$(   - 4231
R0$(   - 4078 4087 4096 4105 4207 4231 4627 4630 4642 4645
R2$(   - 4207 4210
R4     - 4207 4216 4219
R5     - 4207 4216 4219
R6     - 4210 4213 4231
R8$(   - 4270 4309 4330 4339 4396 4408 4624 4633
W4$    - 4120 4315
W6     - 4117 4228 4321
W6$    - 4033 4036 4039 4042 4084 4090 4099 4108 4114 4318
X6     - 4216
X7     - 4216 4564 4570 4576 4603 4609 4615
X8     - 4207 4219 4267 4270
X9     - 4207 4219 4267 4270
X9$(   - 4270 4570 4609 4672 4675 4678 4681 4684 4687 4690 4693 4696 4699 4726
         4729 4732 4735 4738 4741 4744 4747 4750 4753 4756 4759 4786 4789 4792
         4795 4798 4801 4804 4807 4810 4813
```

```
???? DEFFN' 101
       - 4051 4072 4324 4333 4402 4411
???? DEFFN' 118
       - 4129
4030 DEFFN' 124
4207 DEFFN' 125
4267 DEFFN' 126
       - 4483 4486 4489 4492 4495 4498 4501 4507 4510 4513 4516 4519 4525 4528
         4531 4534 4537 4549 4552 4555 4558 4567 4570 4573 4582 4585 4588 4591
         4606 4609 4612
4306 DEFFN' 127
4393 DEFFN' 128
4477 DEFFN' 129
```

4.7.3 ATH2LST3

This section of the 'ATH2LSTP' program contains subroutines 13Ø to 134.

The remainder of this section presents the 'ATH2LST3' program listing and reference maps.

AUTHOR System: Phase 2 - ATH2LST3 - 1 3 0

```
5000 % ## Program > AUTHOR/2   (ATH2LST3)  Version > 2.0  2200 T/VP  780321
5004 REM **
5008 REM **
5012 REM **  Subroutine  ********************************************** 1 3 0
5016 REM **  Abstract >
5020 REM **    To search disk index for data item ID.
5024 REM **  Variables >
5028 REM **    I4    > itm srch flg ! I5     > indx sctr adrs
5032 REM **    R9    > scratch indx ! Q8$()  > index bufr
5036 REM **    I5$   > search ID    ! I1     > item ID
5040 REM **
5044 REM **********************************************
5048  DEFFN ' 130 (I4)                                          : REM ENTRY
5052              BIN(STR(S5$,1,1))=INT(I4/256)                 : REM CMPUT BINARY ID
5056              BIN(STR(S5$,2,1))=I4-INT(I4/256)*256:         REM CMPUT BINARY ID
5060              I4 = 0                                        : REM INIT SUBSCTR INDX
5064  FOR         D0 = 1 TO 10                                  : REM BEG INDX LOOKUP
5068  DATA LOAD   BA T #2, (D0,D9) Q8$()                        : REM LOAD INDEX SCTR
5072  MAT SEARCH  Q8$(),=STR(S5$,1,2) TO R2$() STEP 4           : REM SRCH INDX SCTR
5076  IF          R2$(1) = HEX(0000)          THEN 5096         : REM ITEM NOT FOUND
5080              I4 = INT(FNR(VAL(R2$(1)))/2) + 1              : REM CMPUT SBSCRT INDX
5084              D1 = D0                                       : REM SAVE SCTR ADRS
5088              D2 = I4                                       : REM SAVE SUBSCRT INDX
5092              D0 = 10                                       : REM SET INDX LOOP END
5096  NEXT        D0                                            : REM END OF INDX LOOKUP
5100  RETURN                                                    : REM EXIT
```

```
5104 REM **
5108 REM **
5112 REM **  Subroutine  ********************************************** 1 3 1
5116 REM **  Abstract >
5120 REM **    To load storage buffer from disk given begining
5124 REM **    sector address.
5128 REM **  Variables  >
5132 REM **      R0$    > begning sctr adrs   ! R0 > bufr indx
5136 REM **      R3$    > scrtch indx adrs    ! R2 > indx item flg
5140 REM **      Q0$() > transfer buffer      ! Q6() > bufr flag
5144 REM **      R1$() > storage buffer       ! R2$()> srch rsults
5148 REM **      R1     > item id             !
5152 REM **
5156 REM ***********************************************
5160  DEFFN ' 131 (R0$,R0,I1,E1)                          : REM ENTRY
5164  INIT(20)    R1$()                                   : REM CLR TEXT BUFFER
5168  IF          E1 = 0                        THEN 5264 : REM NULL LOOKUP
5172              R0 = 1                                  : REM SET BUFR INDX
5176  DATA LOAD   BA T #2, (R0$,R3$) Q9$()                : REM LOAD DATA
5180  MAT COPY    Q9$()<1,252> TO R1$()<R0,252>           : REM LOAD BUFFR
5184  IF          Q9$(64) <> HEX(FFFFFFFF)  THEN 5224     : REM NOT END OF BUFR
5188  MAT SEARCH  R1$(), = R5$(2) TO R2$()                : REM SRCH FOR EOI
5192  IF          R2$(1) <> HEX(0000)       THEN 5252     : REM EOI FOUND
5196  GOSUB ' 101 (16,1,1,0)                              : REM POSITION CUSOR
5200  PRINT       " DATA IMMPROPERLY STORED"              : REM PRNT ERR MSG
5204              R8$ = "99"                              : REM SET ERR CODE
5208              R7$ = "Subroutine 105"                  : REM SET ERR SOURCE
5212  GOSUB ' 000                                         : REM ERR PROCESSING
5216  GOTO        5264                                    : REM BRNCH TO EXIT
5220 REM **
5224              R0  = R0 + 252                          : REM INCR BUFR INDX
5228              R0$ = STR(Q9$(64),3,2)                  : REM STOR NXT ADRS
5232  IF          R0  < 252                 THEN 5176     : REM BUFR NOT FULL
5236              E1  = 9                                 : REM SET FLAG
5240              R0  = 252                               : REM SET TO BUF MAX
5244  GOTO        5264                                    : REM BRANCH TO EXIT
5248 REM **
5252              R0 = FNR(VAL(R2$(1)))                   : REM CMPUTE BYTE STORED
5256  MAT COPY    R5$()<1,1> TO R1$()<R0,1>               : REM CLR EOI BYTE
5260              E1 = 0                                  : REM SET FLAG
5264  RETURN                                              : REM EXIT
```

```
5268 REM **
5272 REM **
5276 REM **  Subroutine  ********************************************** 1 3 2
5280 REM **  Abstract >
5284 REM **    To create a block of text form storage buffer in
5286 REM **    to the page buffer.
5288 REM **  Variables >
5292 REM **    R1$() > storage bufer    ! Q3 > begin line
5296 REM **    R2$() > search buffer    ! Q4 > dflt left mrgn
5300 REM **    R5$() > search id's      ! Q5 > dflt right mrgn
5304 REM **    Q6()  > flags            ! Q6 > buffer pointer
5308 REM **    Q2$() > page buffer      ! Q7 > length of word
5312 REM **    R0$   > begin sctr adrs ! R1 > item id
5316 REM **    I0    > spaces to indent!
5320 REM **
5324 REM ********************************************
5328  DEFFN ' 132  (Q3,Q4,Q5,R0$,I0)                          : REM ENTRY
5332               Q6 = 1                                    : REM INIT BUFR INDX
5336               Q7 = 0                                    : REM INIT WORD LENGTH
5340               Q1 = Q3                                   : REM INIT LINE INDX
5344               Q2 = Q4                                   : REM INIT COL INDX
5348  MAT SEARCH   R1$()<Q6,256-Q6+1>,<>R5$(1)TO R2$()       : REM SRCH FOR CHAR
5352  IF           R2$(1) <> HEX(0000)        THEN 5372      : REM IF BUFR NOT EMPTY
5356  IF           E1 <> 9                    THEN 5400      : REM BLOCK DONE
5360  GOSUB ' 107  (E1,I1)                                   : REM LOAD REST OF ITEM
5364               Q6    = 1                                 : REM RESET BUFR INDX
5368  GOTO         5348                                      : REM BRNCH TO SRCH
5369  REM **
5372               Q6 = FNR(VAL(R2$(1))) + Q6 - 1            : REM CMPUT BUFR POS
5376  MAT SEARCH   R1$()<Q6,256-Q6+1>,=R5$(1) TO R2$()       : REM SRCH FOR WORD END
5380               Q7 = FNR(VAL(R2$(1)))                     : REM WORD LENGTH
5384  MAT COPY     R1$()<Q6,Q7> TO R3$()<1,Q7>               : REM EXTRACT WORD
5388  GOSUB ' 133  (Q1,Q4,Q5,Q7,R3$(1),I0)                   : REM LOAD WORD ONTO PAG
5392               Q6 = Q6 + Q7 - 1                          : REM INCR BUFR INDX
5396  GOTO         5348                                      : REM BRNC TO WRD SRCH
5400  RETURN                                                 : REM EXIT
```

```
5404 REM **
5408 REM **
5412 REM **  Subroutine  ********************************************** 1 3 3
5416 REM **  Abstract >
5420 REM **    To format the input word withen margins and
5424 REM **    to control page formatting and storage control.
5428 REM **  Variables >
5432 REM **    Q1 > current row indx ! Q2 > current col indx
5436 REM **    Q4 > left margin      ! Q5 > right margin
5440 REM **    IO > spaces to indent !
5444 REM **
5448 REM ***********************************************
5452  DEFFN ' 133  (Q1,Q4,Q5,Q7,R3$(1),IO)              : REM ENTRY
5456  IF           Q2+Q7-1 <= Q5              THEN 5492 : REM WORD FITS ON LINE
5460               Q1 = Q1 + 1                          : REM INCR LINE INDX
5464               Q2 = Q4 + IO                         : REM SET LEFT MARICN
5468  IF           Q1 < 26                    THEN 5492 : REM NOT LAST LINE
5472               Q1 = 2                               : REM SET TO T-O-P
5476  GOSUB ' 108  (3)                                  : REM PRINT BUFFER
5480  INIT (20)    Q2$()                                : REM CLR PAC BUFR
5484  GOSUB ' 106  (1,1,0,H8$(H8))                      : REM RE-PRINT HEADR
5488  GOSUB ' 106  (1,50,1,HEX(0E))                     : REM CONTINUE MSG
5492  GOSUB ' 106  (Q1,Q2,Q7,R3$(1))                    : REM WRITE WORD
5496               Q2 = Q2 + Q7                         : REM INCR COL INDX
5500  RETURN                                           : REM EXIT
```

```
5504 REM **
5508 REM **
5512 REM **  Subroutine  ********************************************** 1 3 4
5516 REM **  Abstract >
5520 REM **    This routine will load page buffer with a title
5524 REM **    and format the data base item on the page.
5528 REM **  Variables >
5532 REM **    I1 > item ID              ! H8 > title bufr index
5536 REM **    Q1 > page bufr line indx  ! I4 > lookup return indx
5540 REM **    D2 > indx buffer indx     ! Q2$() > page buffer
5544 REM **    Q8$() > disk indx buffer  ! I0 > expand title flag
5548 REM **
5552 REM ***********************************************
5556  DEFFN ' 134 (I0,I1,H8,N1$)                          : REM ENTRY
5560  IF          Q1 < 25                    THEN 5580    : REM IF SPACE ON PAGE
5564  GOSUB ' 108 (3)                                     : REM PRINT PAG ON LPT
5568  INIT (20)   Q2$()                                   : REM CLR PAGE BUFFER
5572              Q1 = 1                                  : REM RESET LINE INDX
5576  REM **
5580  GOSUB ' 130 (I1)                                    : REM PERFORM DSK LOKUP
5584  IF          I4 = 0                     THEN 5616    : REM ITEM NOT FOUND
5588  IF          I0 = 0                     THEN 5596    : REM DO NOT EXPAND TITL
5592  GOSUB ' 106 (Q1,50,1,HEX(0E))                       : REM INSRT EXPAND FLAG
5596  GOSUB ' 106 (Q1,1,0,H8$(H8))                        : REM INSRT HEADING
5600              Q1 = Q1 + 1                             : REM INCR LINE INDEX
5604  GOSUB ' 131 (Q8$(D2+1),R0,11,1)                     : REM FETCH DATA ITEM
5608  GOSUB ' 132 (Q1,11,70,R0$,0)                        : REM INSRT DATA ITEM
5612              Q1 = Q1 + 2                             : REM INCR LINE INDX
5616  RETURN                                              : REM EXIT
```

```
5096  - 5076
5176  - 5232
5224  - 5184
5252  - 5192
5264  - 5168 5216 5244
5348  - 5368 5396
5372  - 5352
5400  - 5356
5492  - 5456 5468
5580  - 5560
5596  - 5588
5616  - 5584
```

```
D0     - 5064 5068 5084 5092 5096
D1     - 5084
D2     - 5088 5604
D9     - 5068
E1     - 5160 5168 5236 5260 5356 5360
H8     - 5484 5556 5596
H8$(   - 5484 5596
I0     - 5328 5388 5452 5464 5556 5588
I1     - 5160 5360 5556 5580 5604
I4     - 5048 5052 5056 5060 5080 5088 5584
N1$    - 5556
Q1     - 5340 5388 5452 5460 5468 5472 5492 5560 5572 5592 5596 5600 5608 5612
Q2     - 5344 5456 5464 5492 5496
Q2$(   - 5480 5568
Q3     - 5328 5340
Q4     - 5328 5344 5388 5452 5464
Q5     - 5328 5388 5452 5456
Q6     - 5332 5348 5364 5372 5376 5384 5392
Q7     - 5336 5380 5384 5388 5392 5452 5456 5492 5496
Q8$(   - 5068 5072 5604
Q9$(   - 5176 5180 5184 5228
R0     - 5160 5172 5180 5224 5232 5240 5252 5256 5604
R0$    - 5160 5176 5228 5328 5608
R1$(   - 5164 5180 5188 5256 5348 5376 5384
R2$(   - 5072 5076 5080 5188 5192 5252 5348 5352 5372 5376 5380
R3$    - 5176
R3$(   - 5384 5388 5452 5492
R5$(   - 5188 5256 5348 5376
R7$    - 5208
R8$    - 5204
S5$    - 5052 5056 5072
```

```
???? DEFFN' 0
       - 5212
???? DEFFN' 101
       - 5196
???? DEFFN' 106
       - 5484 5488 5492 5592 5596
???? DEFFN' 107
       - 5360
???? DEFFN' 108
       - 5476 5564
5048 DEFFN' 130
       - 5580
5160 DEFFN' 131
       - 5604
5328 DEFFN' 132
       - 5608
5452 DEFFN' 133
       - 5388
5556 DEFFN' 134
```

4.7.4 ATH2LST4

This section of the 'ATH2LSTP' program contains subroutines 190 to 191.

The remainder of this section presents the 'ATH2LST4' program listing and reference maps.

```
6000 %## Program > AUTHOR/2  (ATH2LST4)  Version > 2.0 2200 T/VP  780321
6004 REM **
6008 REM **
6012 REM **  Subroutine  ********************************************** 1 9 0
6016 REM **  Abstract >
6020 REM **    This is a general nine track utility package. This
6024 REM **    Routine will perform the following functions >
6028 REM **      1 - <  BSF > - Backspace one File;
6032 REM **      2 - <  BSR > - Backspace one Record (read ready);
6036 REM **      3 - <  BSW > - Backspace one Record (write ready);
6040 REM **      4 - < CLEAN> - Clean local area on tape;
6044 REM **      5 - <  FSF > - Space Forward one File;
6048 REM **      6 - <  FSR > - Space Forward one Record;
6052 REM **      7 - < READ > - Read Record;
6056 REM **      8 - <REREAD> - Backspace and Read Record;
6060 REM **      9 - <REWIND> - Rewind tape to Load point;
6064 REM **     10 - < WEOF > - Write End Of File mark;
6068 REM **     11 - < WGAP > - Write Inter Record Gap;
6072 REM **     12 - < WRITE> - Write Record.
6076 REM **    A scan for I/O errors will be performed according to
6080 REM **    the following index >
6084 REM **      1 - Data Input/Output errors  (7,12,8)
6088 REM **      2 - Position errors           (1,2,5,6,9,4)
6092 REM **      3 - Control Output errors     (10,11,3)
6096 REM **      4 - No error processing
6100 REM **    A error index is returned in variable >  Z3
6104 REM **    Values of Z3 are >       0  -  Fatal Error
6108 REM **                             1  -  Tape at Load Point
6112 REM **                             2  -  Tape is Write Disabled
6116 REM **                             3  -  End of Tape Sensed
6120 REM **                             4  -  End of File Sensed
6124 REM **  Variables >
6128 REM **    Z1 > Function index         ! Z2 > error index
6132 REM **    Z3 > return error index     ! Z4 > transfer buffer size
6136 REM **    Z5 > suppress error mesg    ! Z6 > device table indx
6140 REM **    Z1$() > micro cmnd buffer   ! Z2$() > error mask buffer
6144 REM **    W2$() > i/o buffer          !
6148 REM **
6152 REM ************************************************
```

```
6156  DEFFN ' 190 (Z1,Z2,Z3,Z4,Z5,Z6)                          : REM ENTRY
6160  DIM         Z1$(13)10,Z2$(12)1,Z3$40                     : REM DEFINE WORK BUFRS
6164  DEFFN       Z(Z)=256*VAL(STR(Z1$(Z),9,1))+FNY(Z):        REM DEFINE FUNCTION
6168  DEFFN       Y(Y)=VAL(STR(Z1$(Y),10,1))                   : REM DEFINE FUNCTION
6172  SELECT      TAPE 078                                     : REM SEL TAPE DEVICE
6176    Z1$( 1) = HEX(6CF34400B607)                            : REM DEFINE MICRO CMND
6180    Z1$( 2) = HEX(6CF14400B607)                            : REM DEFINE MICRO CMND
6184    Z1$( 3) = HEX(6CF04400B607)                            : REM DEFINE MICRO CMND
6188    Z1$( 4) = HEX(6CF74400B607)                            : REM DEFINE MICRO CMND
6192    Z1$( 5) = HEX(6CF44400B607)                            : REM DEFINE MICRO CMND
6196    Z1$( 6) = HEX(6CF24400B607)                            : REM DEFINE MICRO CMND
6200    Z1$( 7) = HEX(6CFB4400C222B607)                        : REM DEFINE MICRO CMND
6204    Z1$( 8) = HEX(6CF84400C222B607)                        : REM DEFINE MICRO CMND
6208    Z1$( 9) = HEX(6CF54400B607)                            : REM DEFINE MICRO CMND
6212    Z1$(10) = HEX(6CF94400B607)                            : REM DEFINE MICRO CMND
6216    Z1$(11) = HEX(6CF64400B607)                            : REM DEFINE MICRO CMND
6220    Z1$(12) = HEX(6CFA4400A2068607)                        : REM DEFINE MICRO CMND
6224    Z1$(13) = HEX(11223344556600000000)                    : REM DEFINE MICRO CMND
6228 REM **
6232    Z2$( 1) = HEX(80)     :     Z2$( 2) = HEX(40)          : REM DEFINE ERR MASKS
6236    Z2$( 3) = HEX(20)     :     Z2$( 4) = HEX(10)          : REM DEFINE ERR MASKS
6240    Z2$( 5) = HEX(08)     :     Z2$( 6) = HEX(04)          : REM DEFINE ERR MASKS
6244    Z2$( 7) = HEX(02)     :     Z2$( 8) = HEX(04)          : REM DEFINE ERR MASKS
6248    Z2$( 9) = HEX(02)     :     Z2$(10) = HEX(01)          : REM DEFINE ERR MASKS
6252 REM **
6256  $CIO TAPE   (Z1$(Z1),Z1$(13))   Q2$()<1,24>              : REM EXECUTE FUNCTION
6260 REM **
6264              Z3 = 0                                       : REM CLR RETURN FLAG
6268 REM **
6272 REM **       !   I/O  !   POS  !   CTL  !  NULL  !
6276 REM **       !--------!--------!--------!--------!
6280  ON Z2 GOTO     6292  ,  6324  ,  6312  ,  6352  : REM BRANCH
6284 REM **       !--------!--------!--------!--------!
6288 REM **
6292  GOSUB ' 191 (7,2)                                       : REM RCVRBL DAT ERR
6296  GOSUB ' 191 (8,9)                                       : REM TRANSMIT DAT ERR
6300  GOSUB ' 191 (8,10)                                      : REM DAT OVERFLOW ERR
6304  GOSUB ' 191 (9,10)                                      : REM DAT OVR/UNDR FLOW
6308 REM **
6312  GOSUB ' 191 (7,1)                                       : REM IRRCVRBL DAT ERR
6316  GOSUB ' 191 (7,4)                                       : REM WRITE DISABLED
6320 REM **
6324  GOSUB ' 191 (7,3)                                       : REM BEG OF TAPE
6328  GOSUB ' 191 (7,5)                                       : REM END OF TAPE
6332  GOSUB ' 191 (7,6)                                       : REM END OF FILE
6336  GOSUB ' 191 (7,7)                                       : REM DEV NOT READY
6340  GOSUB ' 191 (8,8)                                       : REM MICRO CMND ERR
6344  IF          Z3 = 3                    THEN 6256         : REM REPROCS RECORD
6348 REM **
6352  RETURN                                                  : REM EXIT
```

```
6356 REM **
6360 REM **
6364 REM **  Subroutine  ********************************************** 1 9 1
6368 REM **  Abstract >
6372 REM **    To mask the GIO register buffer with error
6376 REM **    mask buffer and display error messages if any.
6380 REM **  Variables >
6384 REM **    Z5 > supress non-fatal msgs  ! Z7 > buffer register
6388 REM **    Z8 > error index to check    ! Z4 > transfer bufr size
6392 REM **    Z9 > characters processed    ! Z1$() > micro cmnd buffer
6396 REM **    Z2$() > error mask buffer    ! Z3$ > non-fatal err msg
6400 REM **    Z3 > return error index      ! Z0 > multi volume flag
6404 REM **
6408 REM ***********************************************
6412  DEFFN ' 191 (Z7,Z8)                                  : REM ENTRY
6416  IF          Z3 <> 0                        THEN 6608 : REM PRIVIOUS ERR
6420  IF          Z7 <> 9                        THEN 6456 : REM NOT COUNT REG
6424  GOSUB ' 191 (7,6)                                    : REM EOF TEST
6428  IF          Z3 = 4                         THEN 6608 : REM EOF RECORD
6432  GOSUB ' 191 (7,5)                                    : REM EOT TEST
6436  IF          Z3 = 3                         THEN 6608 : REM EOT ERROR
6440              Z9 = FNZ(13)                             : REM CMPUT TRANSFR CNT
6444  IF          Z4 <> Z9                       THEN 6508 : REM OVR/UNDR FLOW ERR
6448  GOTO        6608                                     : REM BRANCH TO EXIT
6452 REM **
6456  IF          Z8 <> 5                        THEN 6472 : REM NOT EOT TEST
6460  IF          Z1 <> 9                        THEN 6472 : REM NOT REWIND
6464  IF          Z0 = 99                        THEN 6492 : REM PREVIOUS EOT ERR
6468 REM **
6472  AND         (Z2$(Z8),STR(Z1$(13),Z7,1))              : REM MASK REGISTER
6476  IF          Z2$(Z8) = HEX(00)              THEN 6608 : REM NO ERROR
6480 REM **
6484 REM **       ! 01 F ! 02 F ! 03 I ! 04 I ! 05 I  !
6488 REM **       !------!------!------!------!-------!
6492  ON Z8 GOTO    6508 , 6508 , 6552 , 6556 , 6560  ,
                    6568 , 6508 , 6508 , 6508 , 6508  : REM BRANCH
6496 REM **       !------!------!------!------!-------!
6500 REM **       ! 06 I ! 07 F ! 08 F ! 09 F ! 10 F  !
6504 REM **
6508  PRINT       HEX(01070C0C0C);                         : REM POSITION CUSOR
6512  PRINT       "Touch 'CONTINUE' 'RETURN' to Attempt Recovery";
6516 REM **
6520 REM **       ! 01 F ! 02 F ! 03 I ! 04 I ! 05 I  !
6524 REM **       !------!------!------!------!-------!
6528  ON Z8 GOTO    6544 , 6548 , 6552 , 6556 , 6564  ,
                    6568 , 6572 , 6576 , 6580 , 6584  : REM BRANCH
6532 REM **       !------!------!------!------!-------!
6536 REM **       ! 06 I ! 07 F ! 08 F ! 09 F ! 10 F  !
6540 REM **
```

```
6544  STOP        "IRRECOVERABLE DATA ERROR"     : Z=3 : GOTO 6608
6548  STOP        "RECOVERABLE DATA ERROR"       : Z=3 : GOTO 6608
6552  Z3 = 1      : Z3$ = "BEGINING OF TAPE"             : GOTO 6592
6556  Z3 = 2      : Z3$ = "TAPE WRITE DISABLED"          : GOTO 6592
6560  Z3 = 3      : Z3$ = "END OF TAPE"  : Z0 = 99       : GOTO 6508
6564  STOP        "MOUNT FIRST/NEXT TAPE VOLUME..."      : GOTO 6592
6568  Z3 = 4      : Z3$ = "END OF FILE"                  : GOTO 6592
6572  STOP        "TAPE UNIT NOT READY"          : Z=3 : GOTO 6608
6576  STOP        "INVALID MICRO COMMAND SEQUENCE"       : GOTO 6608
6580  STOP        "DATA READ ERROR"              : Z=3 : GOTO 6608
6584  STOP        "DATA OVER/UNDER FLOW ERROR"   : Z=3 : GOTO 6608
6588 REM **
6592  IF          Z5 = 0                    THEN 6608 : REM SUPPRESS MESG
6596  PRINT       HEX(01070C0C0C);                    : REM POSITION CUSOR
6600  PRINT       STR(Z3$,1,40);                      : REM PRINT MSG
6604 REM **
6608  RETURN                                          : REM EXIT
```

AUTHOR System: Phase 2 - ATH2LST4 - # MAP

```
6256  - 6344
6292  - 6280
6312  - 6280
6324  - 6280
6352  - 6280
6456  - 6420
6472  - 6456 6460
6492  - 6464
6508  - 6444 6492 6560
6544  - 6528
6548  - 6528
6552  - 6492 6528
6556  - 6492 6528
6560  - 6492
6564  - 6528
6568  - 6492 6528
6572  - 6528
6576  - 6528
6580  - 6528
6584  - 6528
6592  - 6552 6556 6564 6568
6608  - 6416 6428 6436 6448 6476 6544 6548 6572 6576 6580 6584 6592
```

```
Q2$(   - 6256
Y      - 6168
Z      - 6164 6544 6548 6572 6580 6584
Z0     - 6464 6560
Z1     - 6156 6256 6460
Z1$(   - 6160 6164 6168 6176 6180 6184 6188 6192 6196 6200 6204 6208 6212 6216
         6220 6224 6256 6472
Z2     - 6156 6280
Z2$(   - 6160 6232 6236 6240 6244 6248 6472 6476
Z3     - 6156 6264 6344 6416 6428 6436 6552 6556 6560 6568
Z3$    - 6160 6552 6556 6560 6568 6600
Z4     - 6156 6256 6444
Z5     - 6156 6592
Z6     - 6156
Z7     - 6412 6420 6472
Z8     - 6412 6456 6472 6476 6492 6528
Z9     - 6440 6444
```

```
6156 DEFFN' 190
6412 DEFFN' 191
       - 6292 6296 6300 6304 6312 6316 6324 6328 6332 6336 6340 6424 6432
```

4.7.5 ATH2LST5

This section of the 'ATH2LSTP' program contains subroutines 200 to 212.

The remainder of this section presents the 'ATH2LST5' program listing and reference maps.

```
7000 %## Program > AUTHOR/2  (ATH2LS15)   Version > 2.0  2200 T/VP  780321
7004 REM **
7008 REM **
7012 REM **  Subroutine  ********************************************** 2 0 0
7016 REM **  Abstract >
7020 REM **    This routine will initialize the line printer
7024 REM **    and print a Title page.
7028 REM **  Variables >
7032 REM **    Q2$() > page buffer          ! P0 > page count index
7036 REM **    Q1 >  page bufr line indx  !
7040 REM **
7044 REM ***********************************************
7048  DEFFN ' 200                                          : REM ENTRY
7052  INIT (20)   Q2$()                                   : REM CLR PAGE BUFFER
7056  INIT (20)   Q4$()                                   : REM CLR MENU FLAGS
7060              P0 = 1                                   : REM INIT PAGE INDX
7064  REM **
7068  GOSUB ' 106 (5,5,1,HEX(0E))                         : REM INSRT EXPAND FLAG
7072  GOSUB ' 106 (5,6,0,"AUTHOR System")                 : REM INSRT TEXT
7076  GOSUB ' 106 (10,5,1,HEX(0E))                        : REM INSRT EXPAND FLAG
7080  GOSUB ' 106 (10,6,0,"Phase  II")                    : REM INSRT TEXT
7084  GOSUB ' 106 (15,5,1,HEX(0E))                        : REM INSRT EXPAND FLAG
7088  GOSUB ' 106 (15,6,0,"Data Base Display")            : REM INSRT TEXT
7092  GOSUB ' 106 (20,5,1,HEX(0E))                        : REM INSRT EXPAND FLAG
7096  GOSUB ' 106 (20,6,0,"Version:  2.0")                : REM INSRT TEXT
7100  REM **
7104  GOSUB ' 108 (3)                                     : REM PRINT TITLE PAGE
7108  INIT (20)   Q2$()                                   : REM CLR PAGE BUFFER
7112              Q1 = 1                                   : REM INIT LINE INDEX
7116  RETURN                                              : REM EXIT
```

```
7120 REM **
7124 REM **
7128 REM **  Subroutine  ********************************************** 2 0 1
7132 REM **  Abstract >
7136 REM **    This routine will test the printer for ready status.
7140 REM **  Variables >
7144 REM **    R8$() > i/o micro command buffer !
7148 REM **
7152 REM ***********************************************
7156  DEFFN ' 201                                        : REM ENTRY
7160  $CIO READY  #5 (4100        ,R8$(9))             : REM SEND CTL CHAR
7164  $IF  ON     #5, 7208                             : REM DEVICE READY
7168  GOSUB ' 101 (13,10,0,0)                          : REM POSITION CUSOR
7172  PRINTUSING  7200, " "                            : REM PRINT ERR MSG
7176  $GIO PAUSE  /005(R8$(4),R8$(9))                  : REM ERROR TONE
7180  GOSUB ' 101 (13,10,1,0)                          : REM ERASE ERR MSG
7184  GOTO        7160                                 : REM CONT READY TEST
7188  REM **
7192  REM **                                             OUTPUT IMAGE FORMATS
7196  REM **
7200  %         * * * *  LINEPRINTER IS NOT READY  * * * *       #
7204  REM **
7208  RETURN                                           : REM EXIT
```

```
7212 REM **
7216 REM **
7220 REM **  Subroutine  ********************************************** 2 0 2
7224 REM **  Command    >  SYMBOL
7228 REM **  Function   >
7232 REM **    To allow the entry of Symbol Descriptions.
7236 REM **  Variables >
7240 REM **    R8$() > micro cmnd bufr      ! R9$() > scratch bufr
7244 REM **    L1    > set number           ! M1$   > set number
7248 REM **    L2    > symbol number        ! M2$   > symbol number
7252 REM **    L3    > item number          ! M3$   > set name
7256 REM **    L4    > graphic code         ! M4$   > graphic code
7260 REM **    R2    > input numric val     ! R2$   > inpt alph val
7264 REM **    K1    > crt row indx         ! K2    > crt col indx
7268 REM **    K5    > buffer indx          ! K6    > line number
7272 REM **  Function Definitions >
7276 DEFFN     I(L3) = L1*1000+L2*10+L3    : REM compute symbol item ID
7280 REM **
7284 REM ************************************************
7288  DEFFN ' 202                                        : REM ENTRY
7292  DIM          M1$3,M2$4,M3$40,M4$1                 : REM DEFINE BUFRS
7296               M1$    = "001"                       : REM INIT SBL SET #
7300               M2$    = "0001"                      : REM INIT SBL NUMBER
7304               M3$    = "Symbol Set #xxx"           : REM INIT SBL SET NAM
7308               M4$    = "1"                         : REM INIT GRAPHIC CODE
7312               N2$    = "SETNUMBER"                 : REM INIT FIRST ITEM
7316  INIT(20)     G4$()                                : REM CLR ITEM FLG BUFR
7320               L1     = 0                           : REM INIT SET # INDX
7324               L2     = 0                           : REM INIT SBL # INDX
7328  GOSUB ' 219 (R0)                                  : REM INPUT COMMAND
7332 REM **        ! S NUM  ! SB NUM ! S NAME ! RF NUM  ! SYMBOL !
7336 REM **        !--------!--------!--------!---------!--------!
7340  ON R0 GOTO    7364   , 7368   , 7372   , 7376    , 7380   ,
                    7384   , 7388   , 7392   , 7396    , 7400   : REM BRANCH
7344 REM **        !--------!--------!--------!---------!--------!
7348 REM **        ! DEFFN  ! MEMAID ! SIMSBL ! NEXT    ! LIST   !
7352 REM **
7356  GOTO         7408                                 : REM BRANCH TO EXIT
7360 REM **
7364  GOSUB ' 206 (L1)                    : GOTO 7328 : REM ENTER SET NUMBER
7368  GOSUB ' 207 (L2)                    : GOTO 7328 : REM ENTER SBL NUMBR
7372  GOSUB ' 208 (L1,L2)                 : GOTO 7328 : REM INSRT SET NAME
7376  GOSUB ' 209                         : GOTO 7328 : REM INSRT REFRNCE #
7380  GOSUB ' 210 (1,R0$)                 : GOTO 7328 : REM INSRT SYMBOL
7384  GOSUB ' 211 (FN1(3),20,1,R0$,"MNEM") : GOTO 7328 : REM INSRT DEFINITION
7388  GOSUB ' 211 (FNI(4),21,1,R0$,"SIM1") : GOTO 7328 : REM INSRT MNEMONIC
7392  GOSUB ' 212 (1,R0$)                 : GOTO 7328 : REM INSRT SIM SMBLS
7396  GOSUB ' 213                         : GOTO 7328 : REM INCR COUNTERS
7400  GOSUB ' 214                         : GOTO 7408 : REM LIST ALL SBLS
7404 REM **
7408  RETURN                                           : REM EXIT
```

```
7412 REM **
7416 REM **
7420 REM **  Subroutine  ********************************************** 2 0 3
7424 REM **  Abstract >
7428 REM **    This routine will display a brief description
7432 REM **    of the Phase 2 list procedures.
7436 REM **  Variables >
7440 REM **    RO$() > char input bufr  !
7444 REM **
7448 REM ***********************************************
7452  DEFFN ' 203                                          : REM ENTRY
7456  SELECT       PRINT 005                               : REM SEL DISPLAY MODE
7460  PRINT        HEX(03)                                 : REM CLR CRT SCREEN
7464  PRINTUSING   7548, " "                               : REM PRINT HELP FILE
7468  PRINTUSING   7552, " "
7472  PRINTUSING   7556, " "
7476  PRINTUSING   7560, " "
7480  PRINTUSING   7564, " "
7484  PRINTUSING   7560, " "
7488  PRINTUSING   7568, " "
7492  PRINTUSING   7560, " "
7496  PRINTUSING   7572, " "
7500  PRINTUSING   7576, " "
7504  PRINTUSING   7580, " "
7508  PRINTUSING   7560, " "
7512  PRINT              " "
7516  PRINTUSING   7584, " ";
7520  SELECT       PRINT 405                               : REM SEL DISPLAY MODE
7524  REM **
7528  $GIO CSR OFF/005 (4006     ,R8$(9))                  : REM DISABLE CUSOR
7532  KEYIN        RO$(1), 7592, 7592       : GOTO 7532    : REM RESPONCE WAIT
7536  REM **
7540  REM **                                                 OUTPUT IMAGE FORMATS
7544  REM **
7548 % # This Program alows the items of the Phase II Data Base to
7552 % #      be listed on the Lineprinter.  Items may be listed
7556 % #      individualy or the entire Data Base may be listed.
7560 % #             ------------------------------
7564 % # To list items INDIVIDUALY simpliy type the Proper Command.
7568 % # To list the entire Data Base simpliy type the LIST Command.
7572 % # Since this program constructs pages in memory a page
7576 % #      will not be printed until the page is full or the
7580 % #      program is stopped by entering the ESCAPE Command.
7584 % # To Return to Menu Touch the 'RETURN' Key
7588  REM **
7592  RETURN                                               : REM EXIT
```

```
7596 REM **
7600 REM **
7604 REM **  Subroutine  ********************************************** 2 0 4
7608 REM **  Abstract >
7612 REM **    This routine will alow the entire data base to
7616 REM **    be listed.   (LIST Command)
7620 REM **
7624 REM ***********************************************
7628  DEFFN ' 204                                          : REM ENTRY
7632  GOSUB ' 134 (1,001,4,N1$)                            : REM INSRT TITLE
7636  GOSUB ' 134 (1,002,5,N1$)                            : REM INSRT SUBTITLE
7640  GOSUB ' 134 (1,003,6,N1$)                            : REM INSRT AUTHOR(S)
7644  GOSUB ' 134 (1,004,7,N1$)                            : REM INSRT DATE
7648  GOSUB ' 134 (1,005,8,N1$)                            : REM INSRT ORGNZTION
7652  GOSUB ' 134 (1,011,9,N1$)                            : REM INSRT TRAIN OBJ.
7656  GOSUB ' 134 (1,031,10,N1$)                           : REM INSRT IMPORTANCE
7660  GOSUB ' 134 (1,032,11,N1$)                           : REM INSRT USAGE
7664  GOSUB ' 134 (1,022,13,N1$)                           : REM INSRT TYPE OF SBL
7668  GOSUB ' 134 (1,021,12,N1$)                           : REM INSRT NAME OF USER
7672  GOSUB ' 134 (1,023,14,N1$)                           : REM INSRT REFRNC SOURC
7676  GOSUB ' 134 (1,041,15,N1$)                           : REM INSRT EXAMPLE
7680  GOSUB ' 214                                          : REM INSRT ALL SYMBOLS
7684  GOSUB ' 205                                          : REM PRINT LAST PAGE
7688  RETURN                                               : REM EXIT
```

```
7692 REM **
7696 REM **
7700 REM **  Subroutine  ********************************************** 2 0 5
7704 REM **  Abstract >
7708 REM **    This routine will print the final page onto
7712 REM **    the lineprinter. If underscore scaning was
7716 REM **    selected listing will be reprocessed.
7720 REM **              (ESCAPE  Command)
7724 REM **  Variables >
7728 REM **    W4$ > lpt scan flag      !
7732 REM **
7736 REM ***********************************************
7740  DEFFN ' 205                                          : REM ENTRY
7744  GOSUB ' 108 (3)                                      : REM PRINT LAST PAGE
7748  IF          W4$ <> "TAPE"              THEN 7760     : REM NO SPL SCANING
7752  GOSUB ' 108 (4)                                      : REM PRINT UNDERSCORING
7756  REM **
7760  $GIO TOP    /015 (400C400C400C,R8$(9))               : REM REMOVE PAPER
7764  RETURN                                               : REM EXIT
```

```
7768 REM **
7772 REM **
7776 REM **  Subroutine  ********************************************** 2 0 6
7780 REM **  Abstract >
7784 REM **    To allow the entry of the Set Number.
7788 REM **           (SET NUMBER      Command)
7792 REM **  Variables >
7796 REM **    L1 > symbol set number    ! M1$ > symbol set number <display>
7800 REM **    L2 > symbol number        ! M2$ > symbol number <display>
7804 REM **    N2$ > next command indx   ! R2$ > return input value
7808 REM **
7812 REM ***********************************************
7816  DEFFN ' 206 (L1)                                          : REM ENTRY
7820  PRINT       HEX(03)                                       : REM CLR CONSOL SCREN
7824  GOSUB ' 101 (14,1,0,0)                                    : REM POSITION CUSOR
7828  INIT(2D)    R9$()                                         : REM CREATE DASH LINE
7832  $GIO DASH   /005 (R8$(2),R8$(9))R9$()<1,63>               : REM PRINT LINE
7836  GOSUB ' 101 (1,1,0,0)                                     : REM POSITION CUSOR
7840  PRINT       "Enter Set Number > "                         : REM PRINT PROMPT
7844  GOSUB ' 104 (3,"N",M1$)                                   : REM INPUT SET NUMBR
7848              L1 = R2                                       : REM SAVE SET NUMBR
7852  CONVERT     R2 TO R2$, (###)                              : REM CONV SET NUMBR
7856  GOSUB ' 101 (15,1,0,0)                                    : REM POSITION CUSOR
7860  PRINT       "Set # > ";STR(R2$,1,3)                       : REM PRINT SET NUMBR
7864  IF          STR(R2$,1,3) = M1$          THEN 7872         : REM SAVE SET NUMBR
7868              M2$ = "0001"                                  : REM RESET SBL NUMBR
7872              M1$ = R2$                                     : REM SAVE NEW SET #
7876              N2$ = "SBLNUMBER"                             : REM SET NEXT COMMAND
7880              Q4$(9) = HEX(20)                              : REM CLR ITEM FLG-NEXT
7884  RETURN                                                    : REM EXIT
```

```
7888 REM **
7892 REM **
7896 REM **  Subroutine  ********************************************** 2 0 7
7900 REM **  Abstract >
7904 REM **    To allow the entry of the Symbol Number.
7908 REM **          (SBL NUMBER      Command)
7912 REM **  Variables >
7916 REM **    L1 > symbol set number    ! M1$ > symbol set number <display>
7920 REM **    L2 > symbol number        ! M2$ > symbol number <display>
7924 REM **    N2$ > next command indx   ! R2$ > return input value
7928 REM **
7932 REM ************************************************
7936  DEFFN ' 207 (L2)                                    : REM ENTRY
7940  GOSUB ' 101 (1,1,1,0)                               : REM POSITION CUSOR
7944  PRINT       "Enter Symbol Number > "                : REM PRINT PROMPT
7948  GOSUB ' 104 (4,"N",M2$)                             : REM INPUT SBL NUMBR
7952              L2 = R2                                 : REM SAVE SBL NUMBR
7956  CONVERT     R2 TO M2$, (####)                       : REM CONV SBL NUMBR
7960  GOSUB ' 101 (16,1,0,0)                              : REM POSITION CUSOR
7964  PRINT       "Sbl # > ";M2$                          : REM PRINT SBL NUMBR
7968              N2$ = "SETNAME"                         : REM SET NEXT COMMAND
7972  RETURN                                              : REM EXIT
```

```
7976 REM **
7980 REM **
7984 REM **  Subroutine  ********************************************** 2 0 8
7988 REM **  Abstract >
7992 REM **    This routine will print the current symbol
7996 REM **    set name after verifying that a valid set number
8000 REM **    and symbol number have been entered.
8004 REM **  Variables >
8008 REM **    L1 > set number            ! L2 > symbol number
8012 REM **    N2$ > next command indx    !
8016 REM **
8020 REM ***********************************************
8024  DEFFN ' 208 (L1,L2)                                  : REM ENTRY
8028  GOSUB ' 215 (L1,L2)                                  : REM VERIFIY ID
8032  GOSUB ' 217                                          : REM INSRT HEADER
8036  GOSUB ' 218 (1,RO$)                                  : REM INSRT SET NAME
8040             N2$ = "REFERENCE NUMBER"                  : REM SET NXT COMMAND
8044  RETURN                                               : REM EXIT
```

```
8048 REM **
8052 REM **
8056 REM **  Subroutine  ********************************************** 2 0 9
8060 REM **  Abstract >
8064 REM **    This routine will load the symbol set number
8068 REM **    and symbol number into the page buffer.
8072 REM **  Variables >
8076 REM **    H8$() > header buffer       ! M1$ > set number <display>
8080 REM **    N2$ > next command indx     ! M2$ > sbl number <display>
8084 REM **    Q1 > pag bufr line index    !
8088 REM **
8092 REM ***********************************************
8096  DEFFN ' 209                                        : REM ENTRY
8100              STR(H8$(18),17,3) = M1$                : REM INSRT SET NUMBR
8104              STR(H8$(18),37,4) = M2$                : REM INSRT SBL NUMBR
8108  GOSUB ' 106 (Q1,2,0,H8$(18))                       : REM INSRT HEADING
8112              Q1 = Q1 + 1                            : REM INCR LINE INDX
8116              N2$ = "SYMBOL"                         : REM SET NEXT COMMAND
8120  RETURN                                             : REM EXIT
```

```
8124 REM **
8128 REM **
8132 REM **  Subroutine  ********************************************** 2 1 0
8136 REM **  Abstract >
8140 REM **    This routine will load the symbol and the
8144 REM **    symbol reference code into the page buffer.
8148 REM **  Variables >
8152 REM **    R0 > disk lookup flag          ! R0$ > beging disk location
8156 REM **    Q8$() > disk index buffer      ! D2 > index buffer index
8160 REM **    Q1 > page bufr line index      ! Q2$() > page buffer
8164 REM **    H8$() > heading buffer         ! M4$ > symbol graphic code
8168 REM **    M1$ > symbol set number        ! M2$ > symbol number
8172 REM **
8176 REM ************************************************
8180  DEFFN ' 210 (R0,R0$)                                  : REM ENTRY
8184  IF          R0 = 0                          THEN 8200 : REM IF NULL LOOKUP
8188  GOSUB ' 130 (FNI(2))                                  : REM PERFORM DSK LOKUP
8192              R0$ = Q8$(D2+1)                           : REM STOR BEG LOCATION
8196  REM **
8200  GOSUB ' 131 (R0$,R0,FNI(2),1)                         : REM FETCH SYMBOL
8204              STR(H8$(19),23,2) = "00"                  : REM INSRT SBL CODE
8208  GOSUB ' 106 (Q1,2,0,H8$(19))                          : REM INSRT HEADING
8212  MAT COPY    R1$()<1,R0> TO Q2$()<S6+S3+2,R0>          : REM INSRT SYMBOL
8216              Q1 = Q1 + 1                               : REM INCR LINE INDX
8220              N2$ = "DEFINITION"                        : REM SET NEXT COMMAND
8224  RETURN                                                : REM EXIT
```

```
8228 REM **
8232 REM **
8236 REM **  Subroutine  ********************************************** 2 1 1
8240 REM **  Abstract >
8244 REM **    This routine will load either the definition
8248 REM **    or mnemonic into the page buffer.
8252 REM **  Variables >
8256 REM **    I1 > item ID              ! H8 > heading bufr index
8260 REM **    RO > disk lookup index    ! RO$ > beging disk location
8264 REM **    N2$ > next command index  ! D2 > index buffer index
8268 REM **    Q8$() > disk indx bufr    ! Q1 > page bufr line indx
8272 REM **
8276 REM ***********************************************
8280  DEFFN ' 211 (I1,H8,RO,RO$,N2$)                      : REM ENTRY
8284  IF          RO = 0                        THEN 8300 : REM IF NUL DSK LOKUP
8288  GOSUB ' 130 (I1)                                    : REM PERFORM DSK LOKUP
8292              RO$ = Q8$(D2+1)                         : REM STOR BEG LOCATION
8296  REM **
8300  GOSUB ' 131 (RO$,RO,I1,1)                           : REM FETCH DATA ITEM
8304  GOSUB ' 106 (Q1,2,0,H8$(H8))                        : REM INSRT HEADING
8308              Q1 = Q1 + 1                             : REM INCR LINE INDX
8312  GOSUB ' 132 (Q1,20,70,RO$,0)                        : REM INSRT DATA ITEM
8316              Q1 = Q1 + 1                             : REM INCR LINE INDX
8320  RETURN                                              : REM EXIT
```

```
8324 REM **
8328 REM **
8332 REM **  Subroutine  ********************************************** 2 1 2
8336 REM **  Abstract >
8340 REM **    This routine will load the similar symbols into
8344 REM **    the page buffer.
8348 REM **  Variables >
8352 REM **    RO > disk lookup index      ! RO$ > begining disk location
8356 REM **    Q8$() > disk indx bufr      ! D2 > index buffer index
8360 REM **    N2$ > next command index    ! Q1 > page bufr line index
8364 REM **    H8$() > heading buffer      ! T1 > similar sbl item index
8368 REM **    R9$() > temp text buffer    !
8372 REM **
8376 REM ***********************************************
8380  DEFFN ' 212 (RO,RO$)                                  : REM ENTRY
8384  IF          RO = 0                          THEN 8400 : REM IF NUL DSK LOKUP
8388  GOSUB ' 130 (FN1(5))                                  : REM PERFORM DSK LOKUP
8392              RO$ = Q8$(D2+1)                           : REM STOR DSK LOCATION
8396  REM **
8400  GOSUB ' 131 (RO$,RO,FN1(5),1)                         : REM FETCH DATA ITEM
8404  GOSUB ' 106 (Q1,2,0,H8$(22))                          : REM INSRT HEADING
8408              Q1 = Q1 + 1                               : REM INCR LINE INDX
8412  REM **
8416  FOR         T1 = 1 TO INT(RO/7)                       : REM BEG DISPLAY LOOP
8420  INIT (20)   R9$()                                     : REM CLR TEMP BUFFER
8424  MAT COPY    R1$()<(T1-1)*7+4,4> TO R9$()<5,4>         : REM LOAD TEMP BUFFER
8428  IF          STR(R9$(1),5,4) <> "9999" THEN 8456       : REM NOT END OF LIST
8432  IF          T1 = 1                          THEN 8448 : REM NO SIM SYMBOLS
8436              T1 = INT(RO/7)                            : REM SET TO LOOP END
8440  GOTO        8468                                      : REM BRANCH END OF LOOP
8444  REM **
8448              STR(R9$(1),5,4) = "NONE"                  : REM SET END MSG
8452  REM **
8456  CONVERT     T1 TO STR(R9$(1),1,4), (##.#)             : REM INSRT ITEM NUMBR
8460              STR(R9$(1),4,1) = "    "                  : REM INSRT SPACE
8464  GOSUB ' 106 (Q1,(T1-1)*13+20,13,R9$(1))               : REM INSRT DATA ITEM
8468  NEXT        T1                                        : REM END OF DISPLY LOOP
8472  REM **
8476              Q1 = Q1 + 4                               : REM INCR LINE INDX
8480              N2$ = "NEXT"                              : REM SET NEXT COMMAND
8484  RETURN                                                : REM EXIT
```

```
7160  - 7184
7200  - 7172
7208  - 7164
7328  - 7364 7368 7372 7376 7380 7384 7388 7392 7396
7364  - 7340
7368  - 7340
7372  - 7340
7376  - 7340
7380  - 7340
7384  - 7340
7388  - 7340
7392  - 7340
7396  - 7340
7400  - 7340
7408  - 7356 7400
7532  - 7532
7548  - 7464
7552  - 7468
7556  - 7472
7560  - 7476 7484 7492 7508
7564  - 7480
7568  - 7488
7572  - 7496
7576  - 7500
7580  - 7504
7584  - 7516
7592  - 7532
7760  - 7748
7872  - 7864
8200  - 8184
8300  - 8284
8400  - 8384
8448  - 8432
8456  - 8428
8468  - 8440
```

```
D2     - 8192 8292 8392
H8     - 8280 8304
H8$(   - 8100 8104 8108 8204 8208 8304 8404
I1     - 8280 8288 8300
L1     - 7276 7320 7364 7372 7816 7848 8024 8028
L2     - 7276 7324 7368 7372 7936 7952 8024 8028
L3     - 7276
M1$    - 7292 7296 7844 7864 7872 8100
M2$    - 7292 7300 7868 7948 7956 7964 8104
M3$    - 7292 7304
M4$    - 7292 7308
N1$    - 7632 7636 7640 7644 7648 7652 7656 7660 7664 7668 7672 7676
N2$    - 7312 7876 7968 8040 8116 8220 8280 8480
P0     - 7060
Q1     - 7112 8108 8112 8208 8216 8304 8308 8312 8316 8404 8408 8464 8476
Q2$(   - 7052 7108 8212
Q4$(   - 7056 7316 7880
Q8$(   - 8192 8292 8392
R0     - 7328 7340 8180 8184 8200 8212 8280 8284 8300 8380 8384 8400 8416 8436
R0$    - 7380 7384 7388 7392 8036 8180 8192 8200 8280 8292 8300 8312 8380 8392
         8400
R0$(   - 7532
R1$(   - 8212 8424
R2     - 7848 7852 7952 7956
R2$    - 7852 7860 7864 7872
R8$(   - 7160 7176 7528 7760 7832
R9$(   - 7828 7832 8420 8424 8428 8448 8456 8460 8464
S3     - 8212
S6     - 8212
T1     - 8416 8424 8432 8436 8456 8464 8468
W4$    - 7748
```

AUTHOR System: Phase 2 - ATH2LST5 - ' MAP

```
???? DEFFN' 101
       - 7168 7180 7824 7836 7856 7940 7960
???? DEFFN' 104
       - 7844 7948
???? DEFFN' 106
       - 7068 7072 7076 7080 7084 7088 7092 7096 8108 8208 8304 8404 8464
???? DEFFN' 108
       - 7104 7744 7752
???? DEFFN' 130
       - 8188 8288 8388
???? DEFFN' 131
       - 8200 8300 8400
???? DEFFN' 132
       - 8312
???? DEFFN' 134
       - 7632 7636 7640 7644 7648 7652 7656 7660 7664 7668 7672 7676
7048 DEFFN' 200
7156 DEFFN' 201
7288 DEFFN' 202
7452 DEFFN' 203
7628 DEFFN' 204
7740 DEFFN' 205
       - 7684
7816 DEFFN' 206
       - 7364
7936 DEFFN' 207
       - 7368
8024 DEFFN' 208
       - 7372
8096 DEFFN' 209
       - 7376
8180 DEFFN' 210
       - 7380
8280 DEFFN' 211
       - 7384 7388
8380 DEFFN' 212
       - 7392
???? DEFFN' 213
       - 7396
???? DEFFN' 214
       - 7400 7680
???? DEFFN' 215
       - 8028
???? DEFFN' 217
       - 8032
???? DEFFN' 218
       - 8036
???? DEFFN' 219
       - 7328
```

4.7.6 ATH2LST6

This section of the 'ATH2LSTP' program contains subroutines 213 to 219.

The remainder of this section presents the 'ATH2LST6' program listing and reference maps.

```
9000 %##  Program > AUTHOR/2  (ATH2LST6)   Version > 2.0  2200  T/VP  780321
9004 REM **
9008 REM **
9012 REM **  Subroutine  ********************************************** 2 1 3
9016 REM **  Abstract >
9020 REM **    To Increment symbol counter.
9024 REM **  Variables >
9028 REM **    L1 > set number             ! L2 > symbol number
9032 REM **    M1$ > set number <display> ! M2$ > symbol number <display>
9036 REM **    N2$ > next command index    ! Q4$() > commands used flags
9040 REM **
9044 REM ***********************************************
9048  DEFFN ' 213                                          : REM ENTRY
9052              L2     = L2 + 1                         : REM INCR SBL # INDX
9056  CONVERT     L2 TO M2$, (####)                       : REM CONV TO DISPLY
9060              L1     = 0                              : REM RESET INDEX
9064              L2     = 0                              : REM RESET INDEX
9068              N2$    = "SETNUMBER"                    : REM SET NEXT COMMAND
9072  INIT(20)    Q4$()                                   : REM CLR ITEM FLG BUFR
9076              Q4$(9) = HEX(2A)                        : REM SET ITEM FLG-NEXT
9080  RETURN                                              : REM EXIT
```

```
9084 REM **
9088 REM **
9092 REM **  Subroutine  ********************************************** 2 1 4
9096 REM **  Abstract >
9100 REM **    This routine will list all symbols in the
9104 REM **    symbol table.
9108 REM **  Varialbes >
9112 REM **    GO() > special symbol counts ! Q3$() > symbol table
9116 REM **    T2 > symbol index            !
9120 REM **
9124 REM ***********************************************
9128  DEFFN ' 214                                          : REM ENTRY
9132  REM **
9136  FOR         T2 = 1 TO GO(2)                          : REM BEG PRINT LOOP
9140  GOSUB ' 216 (Q3$(T2,1))                              : REM CMPUT REFRNC #'S
9144  GOSUB ' 217                                          : REM PRINT HEADING
9148  GOSUB ' 218 (0,Q3$(T2,2))                            : REM INSRT SET NAME
9152  GOSUB ' 209                                          : REM INSRT REFRNC #
9156  GOSUB ' 210 (0,Q3$(T2,3))                            : REM INSRT SYMBOL
9160  GOSUB ' 211 (I1,20,0,Q3$(T2,4),N2$)                  : REM INSRT DEFINITION
9164  GOSUB ' 211 (I1,21,0,Q3$(T2,5),N2$)                  : REM INSRT MNEMONIC
9168  GOSUB ' 212 (0,Q3$(T2,6))                            : REM INSRT SIM SYMBOLS
9172  NEXT        T2                                       : REM CONT PRINT LOOP
9176  REM **
9180  RETURN                                               : REM EXIT
```

```
9184 REM **
9188 REM **
9192 REM **  Subroutine  ********************************************** 2 1 5
9196 REM **  Abstract >
9200 REM **    This routine will perform a disk lookup of
9204 REM **    the current symbol ID.  If found will return.
9208 REM **    If not found will ask to re-enter ID.
9212 REM **  Variables >
9216 REM **    L1 > set number                ! L2 > symbol number
9220 REM **    I4 > return lookup flag        ! Q8$() > disk index buffer
9224 REM **    D2 > index buffer index        !
9228 REM **
9232 REM ***********************************************
9236  DEFFN ' 215 (L1,L2)                                  : REM ENTRY
9240  GOSUB ' 130 (FNI(0))                                 : REM PERFORM DSK LOKUP
9244  IF          I4 <> 0                       THEN 9264 : REM ID IS OK
9248  GOSUB ' 206 (L1)                                     : REM ENTER SET NUMBER
9252  GOSUB ' 207 (L2)                                     : REM ENTER SBL NUMBER
9256  GOTO        9240                                     : REM RE-CHECK ID
9260  REM **
9264  GOSUB ' 216 (Q8$(D2+1))                              : REM FETCH REFRNC NUMBR
9268  RETURN                                               : REM EXIT
```

```
9272 REM **
9276 REM **
9280 REM **  Subroutine  ********************************************** 2 1 6
9284 REM **  Abstract >
9288 REM **    This routine will compute the symbol set number
9292 REM **    and symbol number from the symbol reference number
9296 REM **  Variables >
9300 REM **    L1 > symbol set number     ! M1$ > symbol set number
9304 REM **    L2 > symbol number         ! M2$ > symbol number
9308 REM **    RO$ > beging dsk location ! M4$ > symbol graphic code
9312 REM **    RO > size of data item     ! R9$() > temp data buffer
9316 REM **
9320 REM ***********************************************
9324  DEFFN ' 216 (RO$)                                    : REM ENTRY
9328  GOSUB ' 131 (RO$,RO,FNI(0),1)                        : REM FETCH REFENC NUMBR
9332  MAT COPY    R1$()<1,RO> TO R9$()<1,RO>               : REM LOAD TEMP BUFR
9336              M4$ = STR(R9$(1),1,1)                    : REM STOR GRAPHIC CODE
9340              M1$ = STR(R9$(1),2,3)                    : REM STOR SET NUMBR
9344              M2$ = STR(R9$(1),5,4)                    : REM STOR SBL NUMBR
9348  CONVERT     M1$ TO L1                                : REM CMPUT SET NUMBR
9352  CONVERT     M2$ TO L2                                : REM CMPUT SBL NUMBR
9356  RETURN                                               : REM EXIT
```

```
9360 REM **
9364 REM **
9368 REM **  Subroutine  ********************************************** 2 1 7
9372 REM **  Abstract >
9376 REM **    This routine will test current line indx
9380 REM **    and if enough space is remaining on page the
9384 REM **    symbol description heading will be inserted into
9388 REM **    page buffer, else the page will be printed and
9392 REM **    the title inserted on next page.
9396 REM **  Variables >
9400 REM **     Q1 > page bufr line index    ! Q2$() > page buffer
9404 REM **     H8$() > heading buffer       ! M1$ > set number
9408 REM **     M2$ > symbol number          !
9412 REM **
9416 REM ********************************************
9420  DEFFN ' 217                                          : REM ENTRY
9424  IF          Q1 = 14                    THEN 9448 : REM SPACE AVAIABLE
9428  IF          Q1 = 1                     THEN 9448 : REM SPACE AVAIABLE
9432  GOSUB ' 108 (3)                                    : REM PRINT PAGE
9436  INIT (20)   Q2$()                                  : REM CLR PAGE BUFFER
9440              Q1 = 1                                 : REM RESET LINE INDX
9444  REM **
9448              STR(H8$(16),30,3) = M1$                : REM INSRT SET NUMBR
9452              STR(H8$(16),34,4) = M2$                : REM INSRT SBL NUMBR
9456  GOSUB ' 106 (Q1,1,1,HEX(0E))                       : REM INSRT EXPAND FLAG
9460  GOSUB ' 106 (Q1,2,0,H8$(16))                       : REM INSRT HEADING
9464              Q1 = Q1 + 1                            : REM INCR LINE INDX
9468  RETURN                                             : REM EXIT
```

```
9472 REM **
9476 REM **
9480 REM **  Subroutine  ********************************************** 2 1 8
9484 REM **  Abstract >
9488 REM **    This routine will load the name of the symbol
9492 REM **    set into the page buffer.
9496 REM **  Variables >
9500 REM **    RO > disk index lookup flag ! RO$ > begining disk location
9504 REM **    Q8$() > disk index buffer   ! D2 > index buffer index
9508 REM **    Q2$() > page buffer         ! R1$() > storage buffer
9512 REM **    Q1 > page buffer line indx  ! H8$() > heading buffer
9516 REM **
9520 REM ***********************************************
9524  DEFFN ' 218 (RO,RO$)                                  : REM ENTRY
9528  IF          RO = 0                          THEN 9544 : REM NUL DISK LOKUP
9532  GOSUB ' 130 (FN1(1))                                  : REM PERFORM DSK LOKUP
9536              RO$ = Q8$(D2+1)                           : REM STOR BEG LOCATION
9540  REM **
9544  GOSUB ' 131 (RO$,RO,FN1(1),1)                         : REM FETCH SET NAME
9548  GOSUB ' 106 (Q1,2,0,H8$(17))                          : REM INSRT HEADING
9552  MAT COPY    R1$()<1,RO> TO Q2$()<S6+S3+2,RO>          : REM INSRT SET NAME
9556              Q1 = Q1 + 1                               : REM INCR LINE INDX
9560  RETURN                                                : REM EXIT
```

```
9564 REM **
9568 REM **
9572 REM **  Subroutine  ********************************************** 2 1 9
9576 REM **  Abstract >
9580 REM **    To display menu for symbol description input
9584 REM **  Variables >
9588 REM **    Q4$() > item memory flags    ! Q5$() > item key buffer
9592 REM **    R0    > return item value    ! R2$() > srch return bufr
9596 REM **    R2$   > input return value   !
9600 REM **
9604 REM ********************************************
9608  DEFFN ' 219 (R0)                                    : REM ENTRY
9612  GOSUB ' 101 (1,1,13,0)                              : REM CLR CRT SCREEN
9616  SELECT      PRINT 005                               : REM SEL CONSOLE DEV
9620  PRINTUSING  9756, " "                               : REM PRINT MENU
9624  PRINTUSING  9760, " "
9628  PRINTUSING  9764, Q4$(1), Q4$(6)
9632  PRINTUSING  9768, Q4$(2), Q4$(7)
9636  PRINTUSING  9772, Q4$(3), Q4$(8)
9640  PRINTUSING  9776, Q4$(4), Q4$(9)
9644  PRINTUSING  9780, Q4$(5), Q4$(10)
9648  PRINTUSING  9784,         " "
9652  SELECT      PRINT 405                               : REM SEL CONSOLE DEV
9656  REM **
9660  GOSUB ' 101 (10,1,0,0)                              : REM POSITION CUSOR
9664  PRINT       "Enter Command > "                      : REM PRINT INSTRUCTN
9668  GOSUB ' 104 (15,"A",N2$)                            : REM INPUT INSTRUCTN
9672              R0  = LEN(R2$)                          : REM CMPUT CMD LEN
9676  IF          R0 <= 6                       THEN 9684 : REM WITHEN RANGE
9680              R0  = 6                                 : REM SET TO MAX LEN
9684  AND         (STR(R2$,1,R0),DF)                      : REM MASK OUT LWR CASE
9688  MAT SEARCH  Q5$(),=STR(R2$,1,R0) TO R2$() STEP 6:   REM SRCH ID TABLE
9692  IF          R2$(2) <> HEX(0000)           THEN 9700 : REM INVALID COMMAND
9696  IF          R2$(1) <> HEX(0000)           THEN 9724 : REM KEY IS FOUND
9700  GOSUB ' 101 (10,38,0,0)                             : REM POSITION CUSOR
9704  PRINT       "RE-ENTER"                              : REM PRINT ERR MSG
9708  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
9712  GOSUB ' 101 (10,38,1,0)                             : REM ERASE ERR MSG
9716  GOTO        9660                                    : REM RE-ENTER CMMND
9720  REM **
9724              R0 = INT(FNR(VAL(R2$(1)))/6) + 1        : REM CMPUT ITEM INDX
9728  IF          R0 <> 12                      THEN 9736 : REM NOT SPL COMMAND
9732              R0 =  10                                : REM RESET CMND INDX
9736              Q4$(R0) = HEX(2A)                       : REM SET MENU FLAG
9740  GOSUB ' 101 (1,1,12,0)                              : REM ERASE MENU
9744  REM **
9748  REM **                                                OUTPUT IMAGE FORMATS
9752  REM **
9756  %  #                  Command  Summary
9760  %  #  ---------------------------------------------
9764  %     #  SETNUMBER                  #  DEFINITION
9768  %     #  SBLNUMBER                  #  MNEMONIC
9772  %     #  SETNAME                    #  SIMILAR SYMBOLS
9776  %     #  REFERENCE NUMBER           #  NEXT
9780  %     #  SYMBOL                     #  LIST
9784  %                                   #  EXIT
9788  REM **
9792  RETURN                                              : REM EXIT
```

AUTHOR System: Phase 2 - ATH2LST6 - # MAP

9240 - 9256
9264 - 9244
9448 - 9424 9428
9544 - 9528
9660 - 9716
9684 - 9676
9700 - 9692
9724 - 9696
9736 - 9728
9756 - 9620
9760 - 9624
9764 - 9628
9768 - 9632
9772 - 9636
9776 - 9640
9780 - 9644
9784 - 9648

AUTHOR System: Phase 2 - ATH2LST6 - V MAP

```
D2     - 9264 9536
H8$(   - 9448 9452 9460 9548
I1     - 9160 9164
I4     - 9244
L1     - 9060 9236 9248 9348
L2     - 9052 9056 9064 9236 9252 9352
M1$    - 9340 9348 9448
M2$    - 9056 9344 9352 9452
M4$    - 9336
N2$    - 9068 9160 9164 9668
Q0(    - 9136
Q1     - 9424 9428 9440 9456 9460 9464 9548 9556
Q2$(   - 9436 9552
Q3$(   - 9140 9148 9156 9160 9164 9168
Q4$(   - 9072 9076 9628 9632 9636 9640 9644 9736
Q5$(   - 9688
Q8$(   - 9264 9536
R0     - 9328 9332 9524 9528 9544 9552 9608 9672 9676 9680 9684 9688 9724 9728
         9732 9736
R0$    - 9324 9328 9524 9536 9544
R1$(   - 9332 9552
R2$    - 9672 9684 9688
R2$(   - 9688 9692 9696 9724
R8$(   - 9708
R9$(   - 9332 9336 9340 9344
S3     - 9552
S6     - 9552
T2     - 9136 9140 9148 9156 9160 9164 9168 9172
```

```
???? DEFFN' 101
       - 9612 9660 9700 9712 9740
???? DEFFN' 104
       - 9668
???? DEFFN' 106
       - 9456 9460 9548
???? DEFFN' 108
       - 9432
???? DEFFN' 130
       - 9240 9532
???? DEFFN' 131
       - 9328 9544
???? DEFFN' 206
       - 9248
???? DEFFN' 207
       - 9252
???? DEFFN' 209
       - 9152
???? DEFFN' 210
       - 9156
???? DEFFN' 211
       - 9160 9164
???? DEFFN' 212
       - 9168
9048 DEFFN' 213
9128 DEFFN' 214
9236 DEFFN' 215
9324 DEFFN' 216
       - 9140 9264
9420 DEFFN' 217
       - 9144
9524 DEFFN' 218
       - 9148
9608 DEFFN' 219
```

5.0 LEARNING MODULE MAINTENANCE PROGRAMS

The purpose of this program is to create and maintain the learning module. This program will create a learning module based on the data base and the module formats. Once the module has been created and stored on disk the program will allow the user to list and/or edit the learning module in much the same fashion as the data base.

The name of this program is 'AUTHOR/3'. For operating instructions see section 5.Ø in Volume I.

5.1 PROGRAM FLOWCHARTS

Figures 5.1 through 5.11 present program flowcharts which show the major steps of the learning module maintenance program. Figure 5.1 is the high level flow diagram listing the major steps. Each major step is then described in greater detail in the subsequent flow diagrams.

When appropriate the subroutine name or overlay name will be shown in the lower portion of the flowchart element. This indicates that the named routine will perform the function described.

5.2 AUTHOR/3

The 'AUTHOR/3' program is presented in three parts. The 'AUTHOR/3' program contains all utility programs used to create the learning module. The routines contained in the 'AUTHOR/3' program remain in memory throughout the construction process.

5.2.1 ATH1/III

This section of the 'AUTHOR/3' program contains the main control logic for creating the learning module along with program interrupt points. Also contained in this section are subroutines ØØØ to 1Ø4.

The remainder of this section presents the 'ATH1/III' program listing and reference maps.

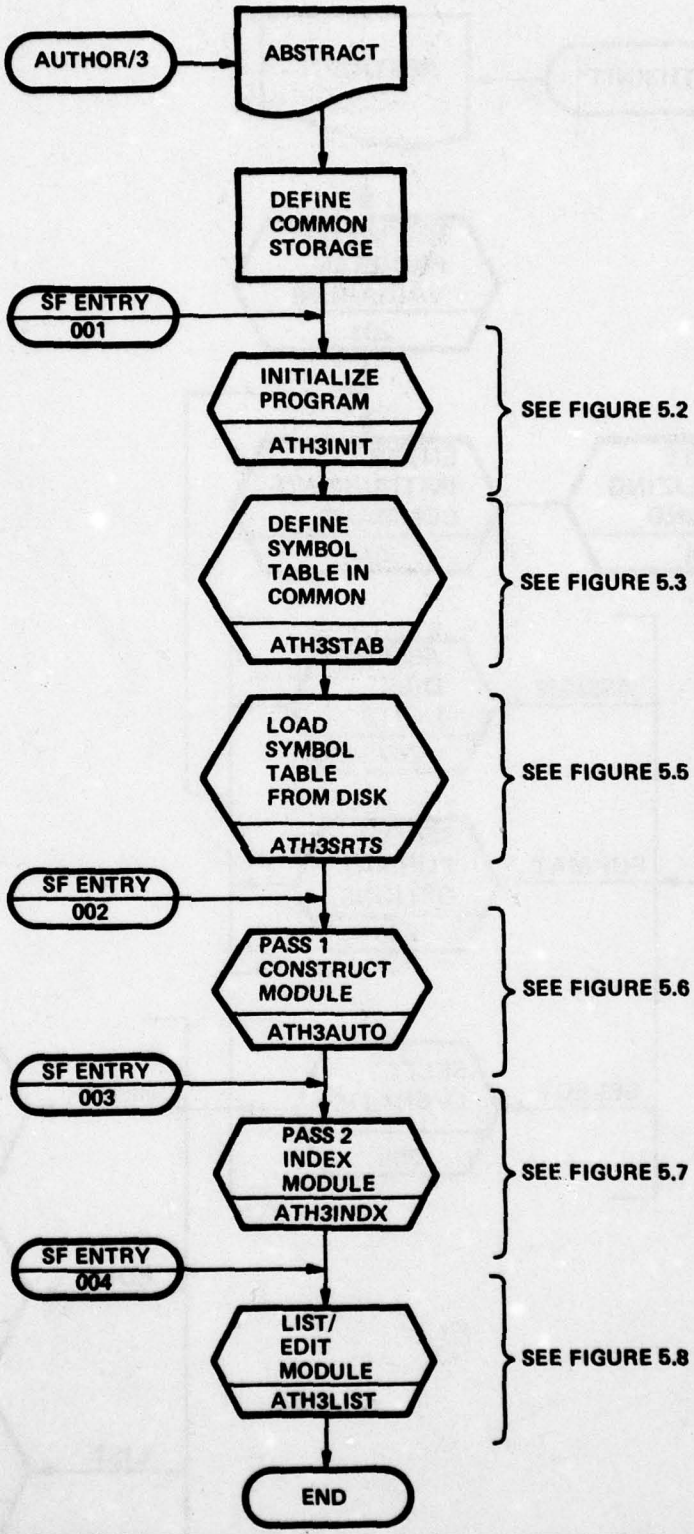


FIGURE 5.1. 'AUTHOR/3' PROGRAM TO CREATE AND MAINTAIN LEARNING MODULE

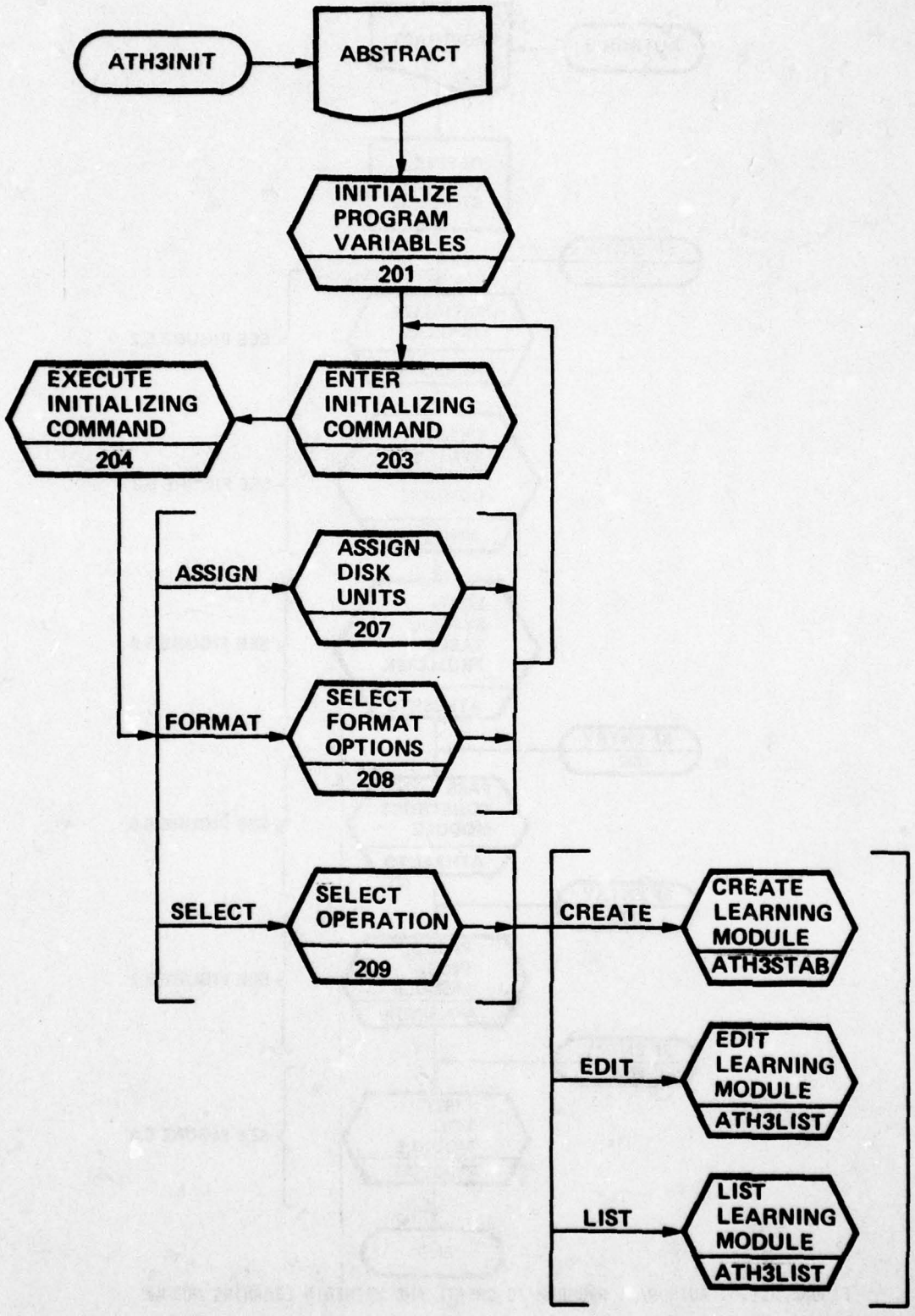


FIGURE 5.2. 'ATH3INIT' PROGRAM TO INITIALIZE LEARNING MODULE MAINTENANCE

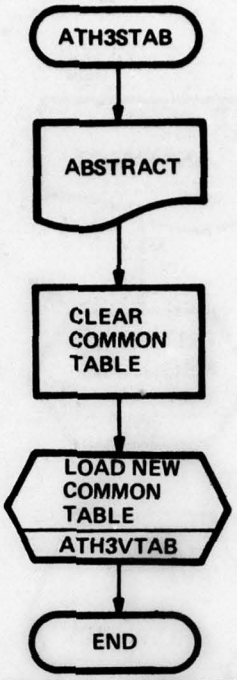


FIGURE 5.3. 'ATH3STAB' PROGRAM TO LOAD SYMBOL TABLE INTO COMMOM

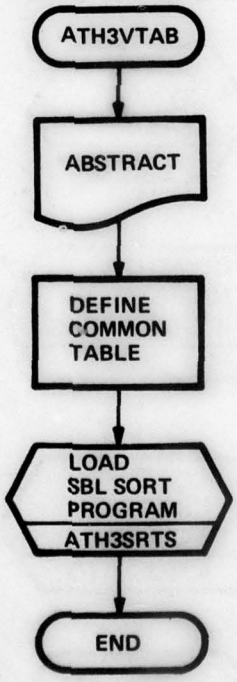


FIGURE 5.4. 'ATH3VTAB' PROGRAM TO LOAD SYMBOL SORT PROGRAM

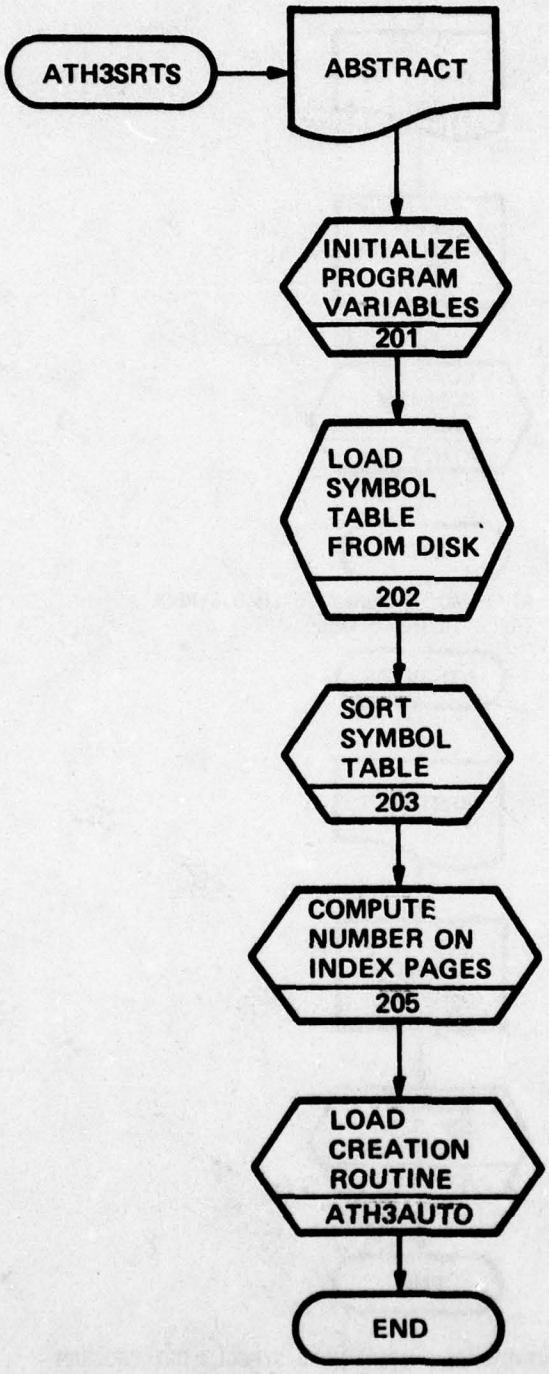


FIGURE 5.5. 'ATH3SRTS' PROGRAM TO LOAD AND SORT SYMBOL TABLE

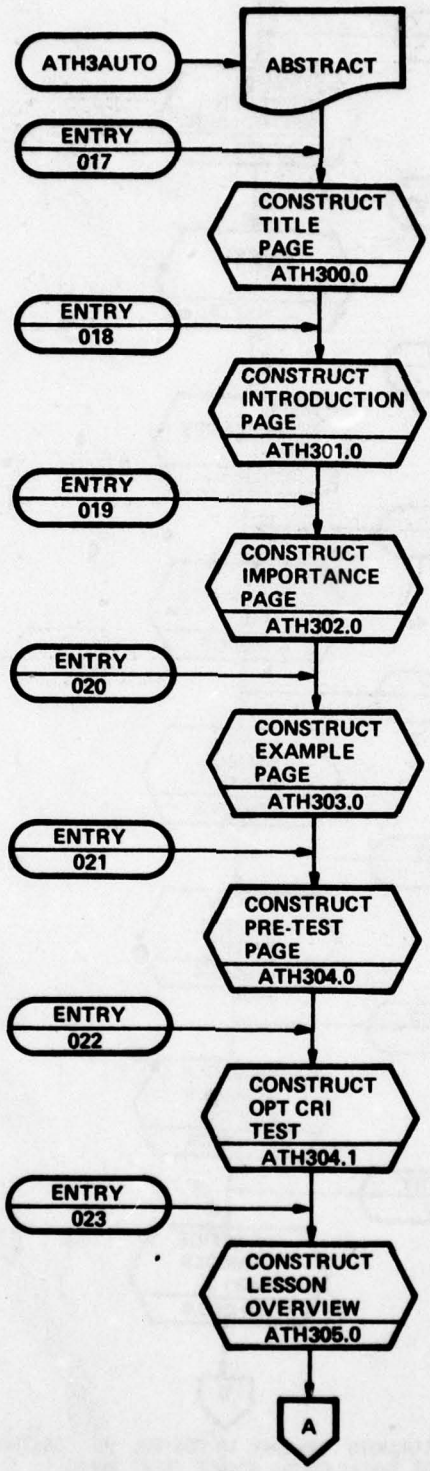


FIGURE 5.6. 'ATH3AUTO' PROGRAM TO CONTROL THE CONSTRUCTION OF THE LEARNING MODULE

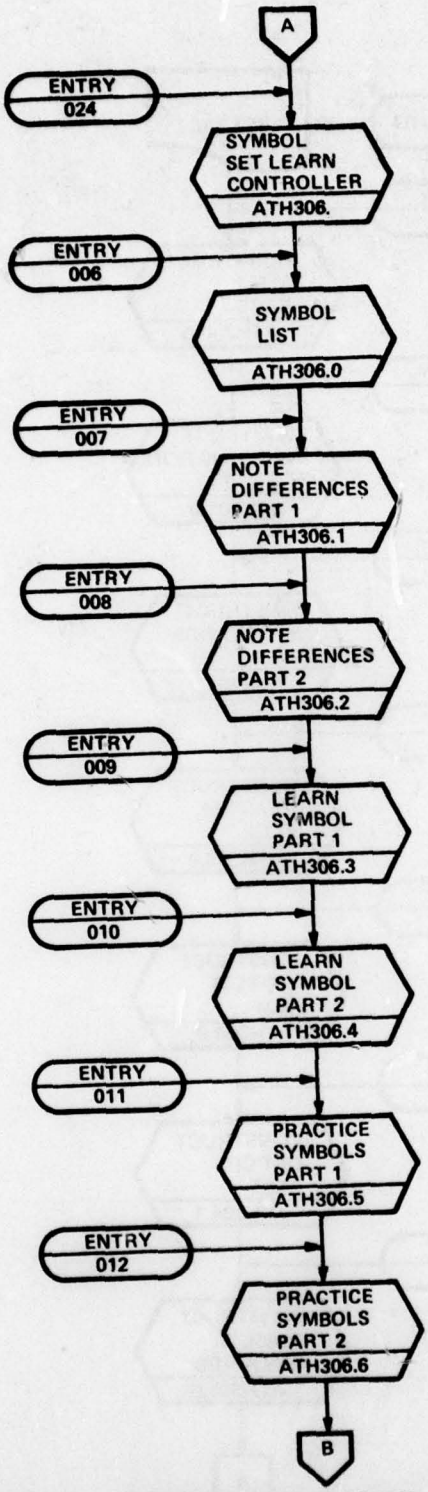


FIGURE 5.6. 'ATH3AUTO' PROGRAM TO CONTROL THE CONSTRUCTION OF THE LEARNING MODULE (CONTINUED)

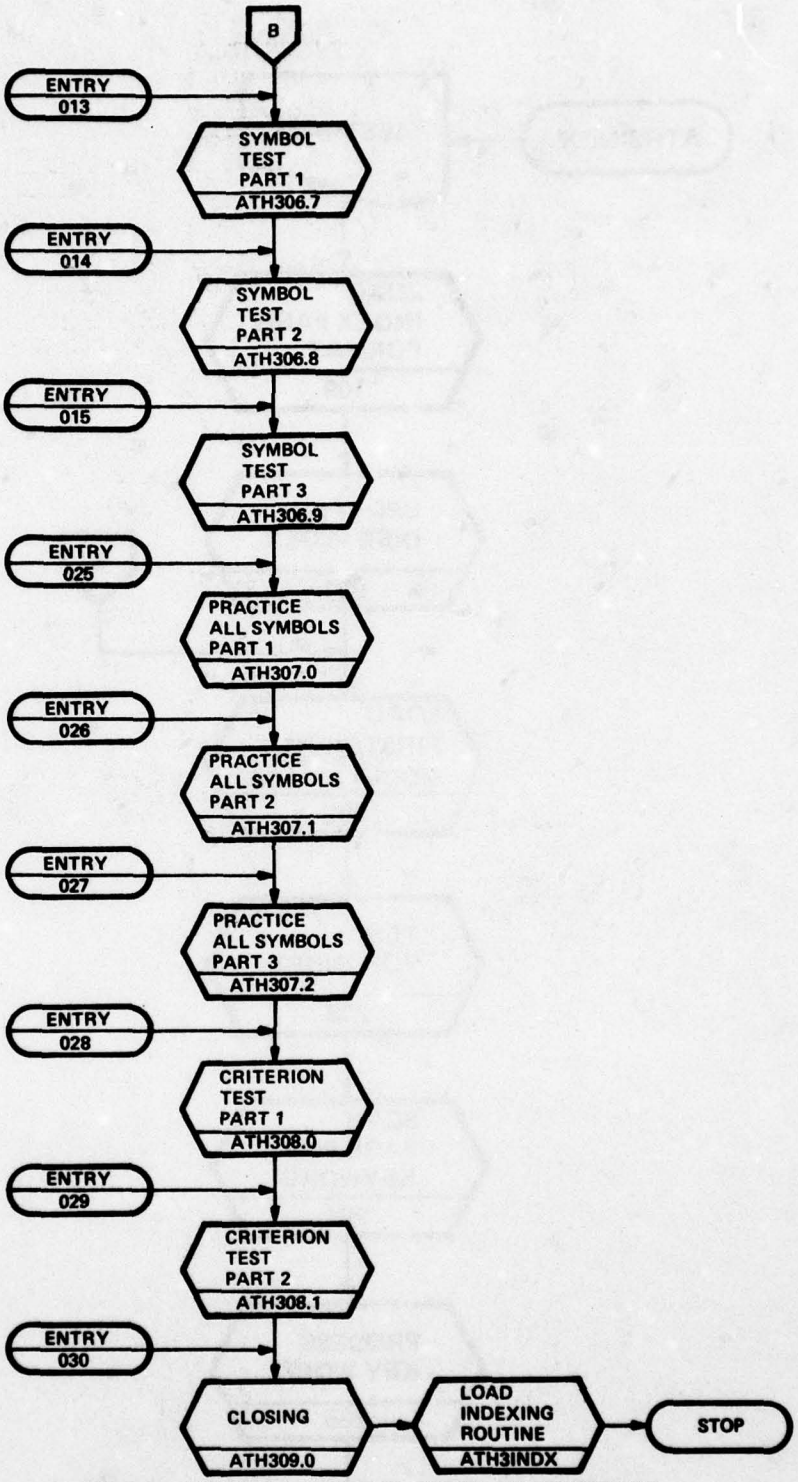


FIGURE 5.6. 'ATH3AUTO' PROGRAM TO CONTROL THE CONSTRUCTION OF THE LEARNING MODULE (CONTINUED)

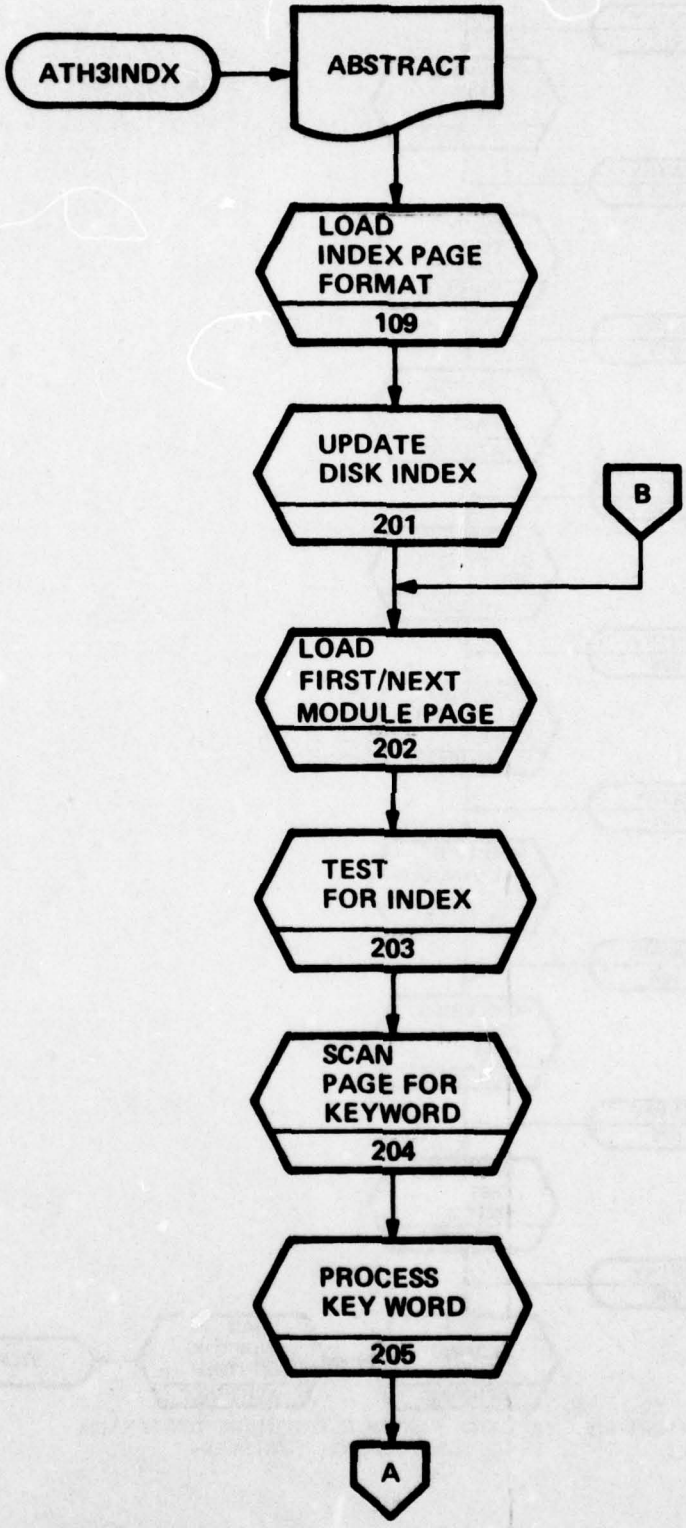


FIGURE 5.7. 'ATH3INDX' PROGRAM TO INDEX THE LEARNING MODULE

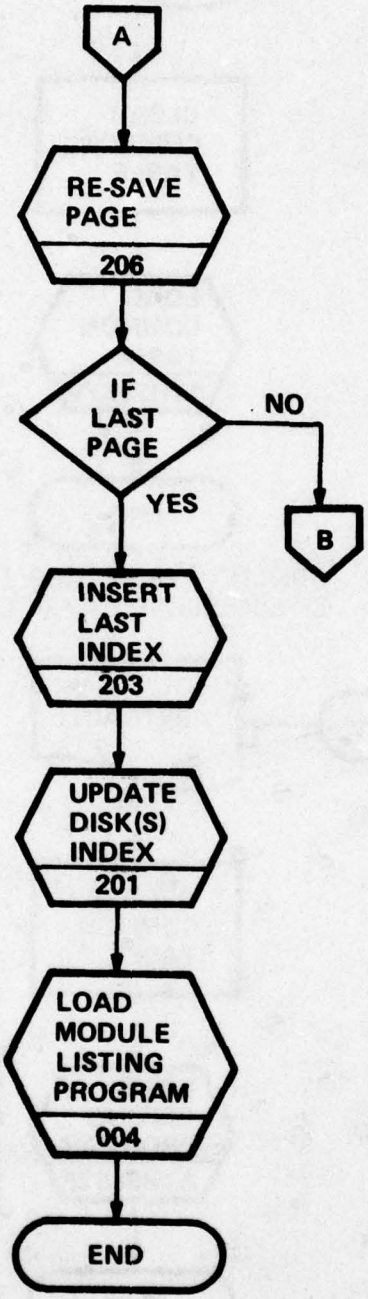


FIGURE 5.7. 'ATH3INDX' PROGRAM TO INDEX THE LEARNING MODULE (CONTINUED)

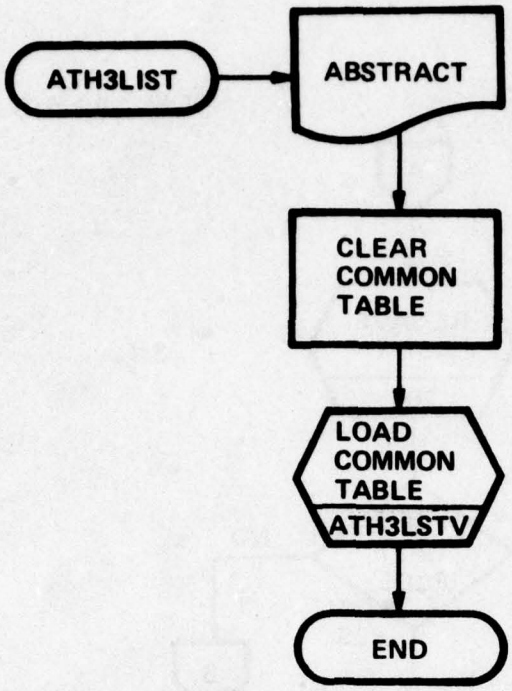


FIGURE 5.8. 'ATH3LIST' PROGRAM TO INITIALIZE LIST PROCESSING BY LOADING NECESSARY COMMON VARIABLES

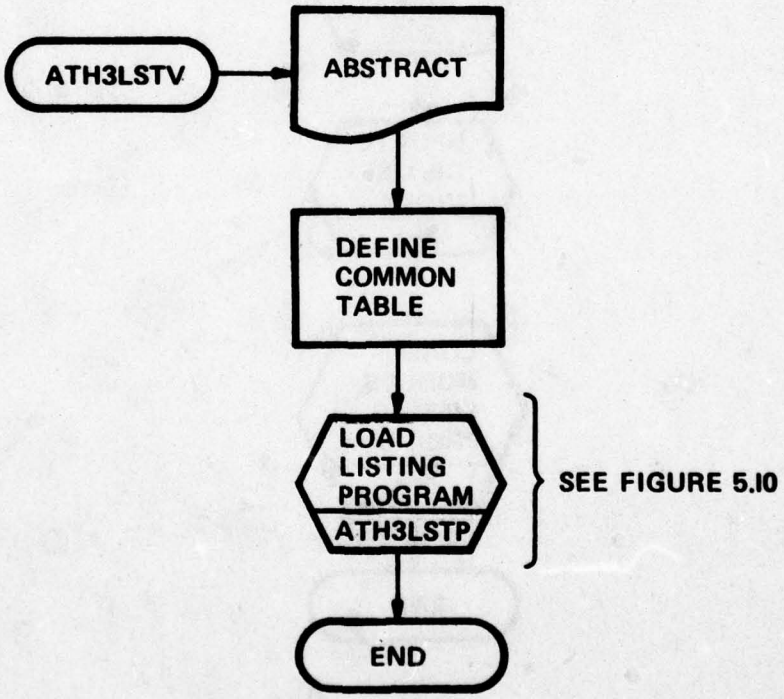


FIGURE 5.9. 'ATH3LSTV' PROGRAM TO LOAD LISTING ROUTINES

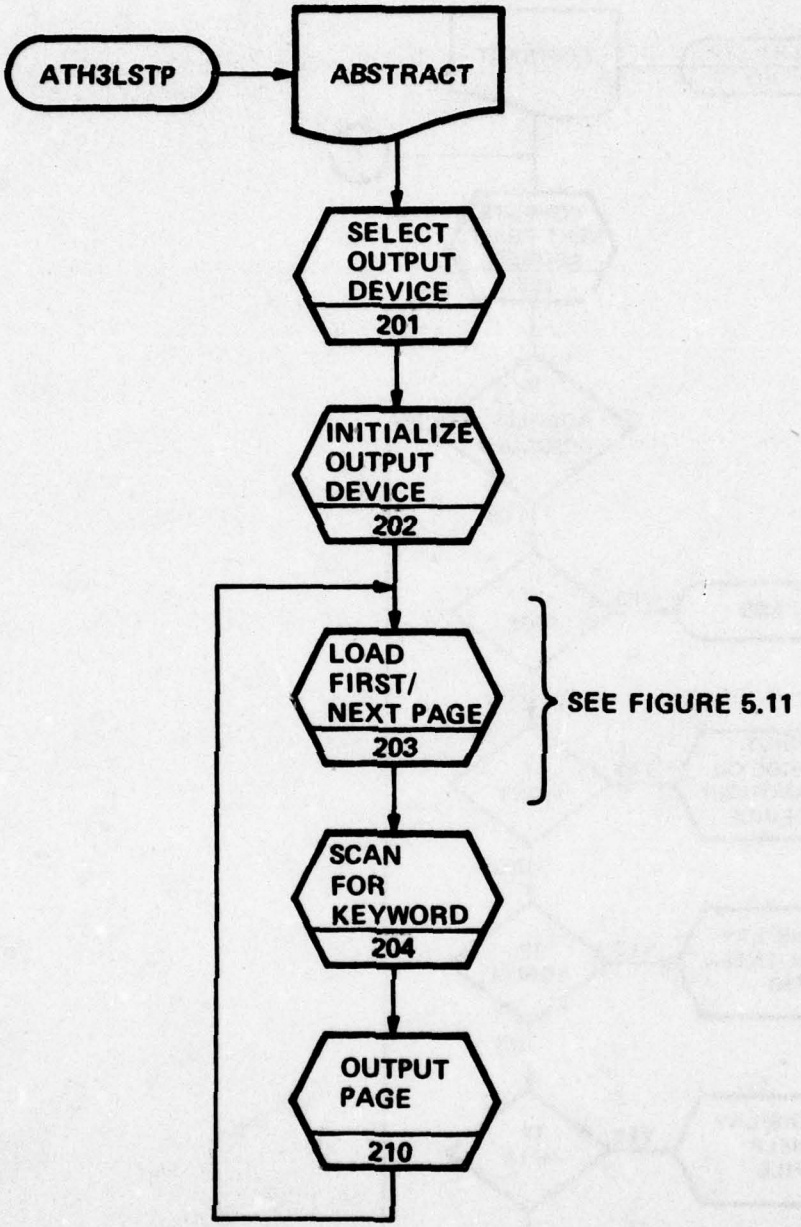


FIGURE 5.10. 'ATH3LSTP' PROGRAM FOR LISTING THE LEARNING MODULE

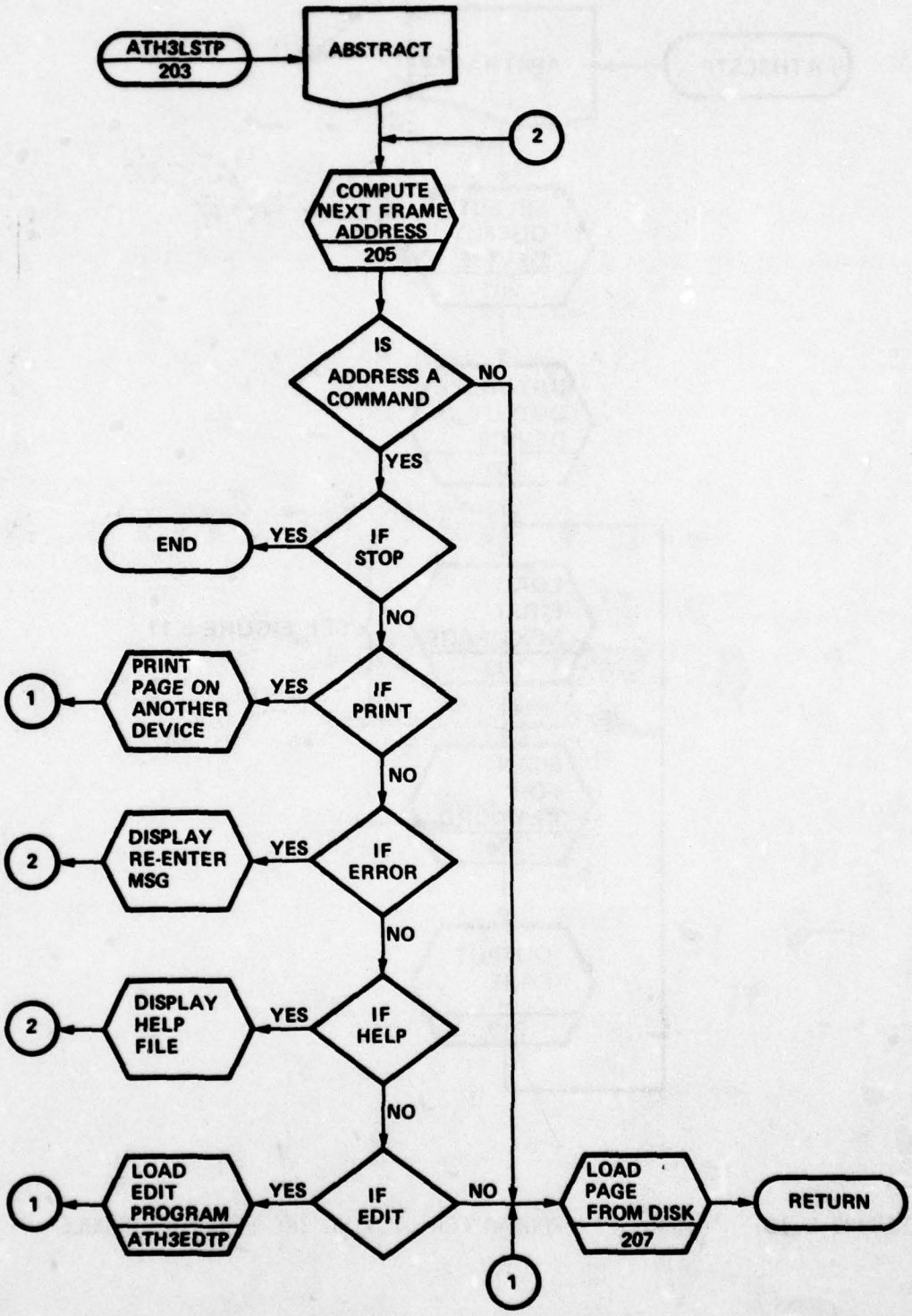


FIGURE 5.11. ''ATH3LSTP' ROUTINE 203--PROGRAM ROUTINE FOR THE RECALL AND EDIT OF LEARNING MODULE

AUTHOR System: Phase 3 - ATH1/III - CNTRL

```
100  REM ** Program > AUTHOR/3  (ATH1/III)  Version > 2.0 2200 T/VP  780321
104  REM **  Abstract  ************************************************************
108  REM **                                                                      **
112  REM **    Phase III will construct from the Phase II Data                   **
116  REM **    Base a Learning Package suitable for output to                    **
120  REM **    Microfiche.  The Learning Package to be                           **
124  REM **    Constructed is for Symbol Learning.                               **
128  REM **                                                                      **
132  REM **  Program Logic                                                       **
136  REM **    Phase III will alow for three modes of operation                  **
140  REM **          -  Construction;                                            **
144  REM **          -  Edit Processing;                                         **
148  REM **          -  List Processing.                                         **
152  REM **                                                                      **
156  REM **      i. The Construction Phase will construct the                    **
160  REM **         entire Symbol Learning Package and                           **
164  REM **         automatically dispose a copy of the output                   **
168  REM **         file to the high speed line printer.                         **
172  REM **     ii. The Edit Phase alows the editing of any                      **
176  REM **         frame of output.                                             **
180  REM **    iii. The List Phase alows the output disk to be                   **
184  REM **         disposed to one of several output devices.                   **
188  REM **         current capabilites alow for output to -                     **
192  REM **          TTY - high quailty output writer DEV <211>                  **
196  REM **          CRT - cathode ray device         DEV <405>                  **
200  REM **          LPT - high speed line printer    DEV <215>                  **
204  REM **                                                                      **
208  REM ************************************************************************
212  REM **
216  REM ***********************************************   DEFINE PROG STROAGE
220  REM **
224  COM       R4$(20)16,R5$(10)1,R6$(16)1,R7$(64)1     : REM GENERAL REGISTERS
228  COM       R8$(10)24,R9$(1)64                       : REM GENERAL REGISTERS
232  COM       B0,B1,B2,B3,B4,B8,B9,B1$1,B1$(20)1       : REM BIT CONTROL
236  COM       D1,D2,D3,D4,D5,D4$4,D5$2                 : REM OUTPUT PAG INDX
240  COM       D6$2,D7$4,D8$2,D9$2                      : REM OUTPUT PAG INDX
244  COM       K1,K2,K3,K4,K5$(1)64                     : REM CUSOR CONTROL
248  COM       L1$25,L2$8,L3$4                          : REM PROGRAM LOADING
252  COM       W1,W2,W3,W4,W5,W6,W1$2                   : REM FICHE PAG INDX
256  COM       U1$(15)1,U0,U1                           : REM SYMBOL INDXING
260  COM       R0,R1,R2,R3,R4,R5,R6,R7,R8,R9            : REM GENERAL REGISTERS
264  COM       R0$2,R1$20,R2$64,R3$2,R4$2,R5$2          : REM GENERAL BUFFERS
268  COM       R6$64,R7$64,R8$2,R9$64,E1$4              : REM GENERAL BUFFERS
272  COM       R0$(1)1,R2$(2)2,R3$(1)64                 : REM GENERAL BUFFERS
276  COM       S1,S2,S3,S4,S5,S4$(1)64,S8$(1)4,S9$(1)4  : REM BUFR STORAGE INDX
280  COM       Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9            : REM GENERAL REGISTERS
284  COM       Q0(12),Q0$(10)1,Q1$(128)2,Q2$(16,64)1    : REM GENERAL BUFFERS
288  COM       Q4$(50)2,Q5$(10)3,Q6$(10)2               : REM GENERAL BUFFERS
292  COM       Q7$(75,2)1                               : REM GENERAL BUFFERS
296  DIM       Q9$(64)4,R1$(1)1                         : REM SCRATCH BUFFER
300  %##            END OF COMMON STORAGE AREA          ##
```

```
304  INIT          (0A)R6$() :  R6$(1) = HEX(01)       : REM INTALIZE CUSOR CTL
308  INIT          (09)R7$() :  R7$(1) = HEX(0D)       : REM INTALIZE CUSOR CTL
312  DEFFN         R(R)= 256*R + VAL(STR(R2$(1),2,1))  : REM FUNCTION R(R)
316  DEFFN         W(W)= 256*W + VAL(STR(W1$,2,1))     : REM FUNCTION W(W)
320  REM **                                                 
324  REM ********************************************  PROGRAM OVERLAY CONTROL
328  REM **
332  SELECT   PRINT 405                                : REM DEFLT OUTPUT DEV
336  DEFFN '  001                                      : REM ENTRY POINT 001
340  GOSUB '  102  ("Initializing","ATH3INIT","4000") : REM LOAD PROGRAM
344  DEFFN '  002                                      : REM ENTRY POINT 002
348  GOSUB '  102  ("Construction","ATH3AUTO","4000") : REM LOAD PROGRAM
352  DEFFN '  003                                      : REM ENTRY POINT 003
356  GOSUB '  102  ("Indexing","ATH3INDX","4000")      : REM LOAD PROGRAM
360  DEFFN '  004                                      : REM ENTRY POINT 004
364  GOSUB '  102  ("Listing","ATH3LIST","1500")       : REM LOAD PROGRAM
368  DEFFN '  005                                      : REM ENTRY POINT 005
372  GOSUB '  102  ("Editing","ATH3LIST","1500")       : REM LOAD PROGRAM
```

```
376  REM **
380  REM **
384  REM **  Subroutine  ********************************************** 0 0 0
388  REM **  Function   >
392  REM **    Program Interrupt  / Error Recovery
396  REM **  Variables  >
400  REM **   R0$() > entry buffer  ! R8$ > error code
404  REM **                         ! R7$ > line number
408  REM ************************************************
412  DEFFN ' 000                                                  : REM ENTRY NORMAL
416  ON ERROR      E1$,R7$                          GOTO 420      : REM ENTRY ERROR
420  GOSUB ' 101   (13,1,2,40)                                   : REM POSITION CUSOR
424  PRINT         HEX(0D0A)                                      : REM POSITION CUSOR
428  PRINT         "Error > ";E1$;HEX(0D0A)                       : REM PRINT ERR MODE
432  PRINT         "Occured > ";R7$;                              : REM PRINT LOCATION
436  IF            R8$ = "99"                        THEN 448     : REM ERR MESSAGE TYPED
440  GOSUB ' 101   (16,1,1,40)                                    : REM POSITION CUSOR
444  PRINT         "ERROR CORRECTION REQUESTED"                   : REM ERR MESSAGE
448  GOSUB ' 101   (1,1,5,40)                                     : REM POSITION CUSOR
452  PRINT         "A U T H O R   S Y S T E M   III"              : REM TITLE LINE
456  PRINT         HEX(0D0A);"END PROGRAM EXECUTION"              : REM SUBTILE LINE
460  PRINT         HEX(0D0C)                                      : REM POSITION CUSOR
464  $GIO TONE     /005 (R8$(4),R8$(9))                           : REM PAUSE TONE
468  END                                                          : REM END PROGRAM
```

```
472 REM **
476 REM **
480 REM **  Subroutine  ********************************************** 0 9 1
484 REM **  Abstract >
488 REM **    To Wait for an Input Responce.
492 REM **
496 REM ***********************************************
500  DEFFN ' 091                                          : REM ENTRY
504  GOSUB ' 101 (15,1,1,0)                               : REM POSITION CUSOR
508  PRINTUSING  560, " "                                 : REM PRINT INSTRUCTNS
512 REM **
516  KEYIN       R0$(1), 532, 532                         : REM KEYBOARD INPUT
520  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
524  GOTO        516                                      : REM CONTINUE WAIT
528 REM **
532  KEYIN       R0$(1), 544, 544                         : REM KEYBOARD INPUT
536  $GIO WAIT   /005 (R8$(1),R8$(9))                     : REM CHAR WAIT
540  GOTO        532                                      : REM CONTINUE WAIT
544  GOSUB ' 101 (15,1,1,0)                               : REM ERASE PROMPT
548 REM **
552 REM **                                                  OUTPUT IMAGE FORMATS
556 REM **
560  % Touch 'RETURN' to Stop Tone...Second 'RETURN' to Continue #
564 REM **
568  RETURN                                               : REM EXIT
```

```
572  REM **
576  REM **
580  REM **  Subroutine  ********************************************** 1 0 0
584  REM **  Abstract  >
588  REM **    To load a string into the page buffer and
592  REM **    display the string on the CRT
596  REM **  Variables >
600  REM **    S1    > row position    ! S2    > col position
604  REM **    S3    > string length   ! S4$()> string
608  REM **    S5    > print flag      ! Q2$()> page buffer
612  REM **    K1    > crt row pos     ! K2    > crt col pos
616  REM **
620  REM ************************************************
624  DEFFN ' 100   (S1,S2,S3,S4$(1),S5)                    : REM ENTRY
628  IF            S3 <> 0                        THEN 636  : REM STR LEN NOT 0
632                S3 =  LEN(S4$(1))                       : REM SET STR LEN
636                S6 = (S1-1)*64 + S2                     : REM SCALE ROW & COL
640  MAT COPY      S4$() <1,S3> TO Q2$() <S6,S3>           : REM LOAD INTO BUFR
644                K1 = S1 - INT((S1-1)/16)*16             : REM SCALE CRT ROW #
648  IF            S5 > 0                         THEN 660  : REM NO OUTPUT TO CRT
652  GOSUB ' 101   (K1,S2,0,0)                             : REM POSITION CUSOR
656  $GIO PRINT    /005(A000,R8$(9))S4$()<1,S3>            : REM OUTPUT TO CRT
660  RETURN                                                : REM EXIT
```

```
664  REM **
668  REM **
672  REM       Subroutine > ********************************************** 1 0 1
676  REM       Function   >
680  REM        Cusor Address and Screen erase
684  REM       Varaibles  >
688  REM        K1     > row position !  K3     > lines to erase
692  REM        K2     > col position !  K4     > char(s) erase
696  REM        R6$()  > cusor vert   !  R7$()  > cusor horz
700  REM        K5,K5$() > line ers   !  R8$()  > micro cmnd bufr
704  REM **********************************************
708  DEFFN '  101   (K1,K2,K3,K4)                                  : REM ENTRY
712  IF             K3 = 0                          THEN 756      : REM ERASE NO LINE
716  IF             K4 <> 0                         THEN 724      : REM CHAR TO ERS SPCFD
720                 K4 = 65 - K2                                  : REM ERS REST OF LINE
724  FOR            K5 = 1 TO K3                                  : REM BEGIN LOOP K5
728  IF             K5 <> 1                         THEN 740      : REM NOT FIRST LINE
732  $GIO ROW       /005(A000  ,R8$(9))R6$()<1,K1>                : REM POSITION CUSOR
736                 GOTO 744                                      : REM BRANCH TO POS COL
740  $GIO ROW       /005(A000  ,R8$(9))R6$()<2,1>                 : REM POSITION CUSOR LF
744  $GIO COL       /005(A000  ,R8$(9))R7$()<1,K2>                : REM POSITION CUSOR
748  $GIO ERASE     /005(A000  ,R8$(9))K5$()<1,K4>                : REM ERASE TEXT
752  NEXT           K5                                            : REM CONTINUE LOOP K5
756  $GIO ROW       /005(A000  ,R8$(9))R6$() <1,K1>               : REM REPOSITION CUSOR
760  $GIO COL       /005(A000  ,R8$(9))R7$() <1,K2>               : REM REPOSITION CUSOR
764  RETURN                                                       : REM EXIT
```

```
768  REM **
772  REM **
776  REM       Subroutine > ********************************************** 1 0 2
780  REM       Function   >
784  REM         To load an over lay Routine.
788  REM       Variables  >
792  REM         R1$ > description        !  R3 > begining line
796  REM         K2$ > overlay name       !
800  REM **********************************************
804  DEFFN '  102  (L1$,L2$,L3$)                              : REM ENTRY
808  PRINT         HEX(03)                                    : REM CLEAR CRT
812  PRINT         "A U T H O R   S Y S T E M"                : REM PRINT TITLE
816  PRINT         HEX(0D0A);"      P H A S E   III"          : REM PRINT SUBTITLE
820  GOSUB '  101  (5,10,0,0)                                 : REM POSITION CUSOR
824  PRINT    "System Now Loading >>> ";L1$;" Routine":       REM PRINT DESCRIPTION
828  GOSUB '  101  (7,10,0,0)                                 : REM POSITION CUSOR
832  PRINT         "Program being Loaded >>> ";L2$            : REM PRINT PROC NAME
836  IF            L3$ <> "0100"                THEN 844      : REM LOAD AT LINE 0100
840  LOAD          DC T #1, L2$ 100
844  IF            L3$ <> "1500"                THEN 852      : REM LOAD AT LINE 1500
848  LOAD          DC T #1, L2$ 1500
852  IF            L3$ <> "4000"                THEN 860      : REM LOAD AT LINE 4000
856  LOAD          DC T #1, L2$ 4000
860  IF            L3$ <> "6000"                THEN 868      : REM LOAD AT LINE 6000
864  LOAD          DC T #1, L2$ 6000
868  IF            L3$ <> "7000"                THEN 876      : REM LOAD AT LINE 7000
872  LOAD          DC T #1, L2$ 7000
876  IF            L3$ <> "9000"                THEN 884      : REM LOAD AT LINE 9000
880  LOAD          DC T #1, L2$ 9000
884                R8$ = "00"                                 : REM SET ERR MODE
888                R7$ = "Subroutine 102"                     : REM SET ERR SOURCE
892  GOSUB ' 000                                             : REM ERR PROCESSING
896  RETURN                                                  : REM EXIT
```

```
900  REM **
904  REM **
908  REM       Subroutine > ********************************************  1 0 3
912  REM       Function   >
916  REM         To perform the following conversions;
920  REM            -<dec> to <hex> page numbers      < 01 >
924  REM            -<hex> to <dec> page numbers      < 02 >
928  REM            -add  n  pages to <hex> number    < 03 >
932  REM            -add  n  pages to <dec> number    < 04 >
936  REM       Varaibles  >
940  REM         W1$ > <hex> page number ! W1 > <dec> fch pag #
944  REM         W2  > <dec> fiche row # ! W3 > <dec> fch col #
948  REM         W4  > num of pags to add! W5 > converison indx
952  REM         Q0(6) > max num of rows ! Q0(7) > max num cols
956  REM         Q0(5) > max num of pages!
960  REM ********************************************
964  DEFFN '  103  (W1$,W1,W2,W3,W4,W5)                  : REM ENTRY
968  REM **
972  REM **        ! DEC->HEX ! HEX->DEC ! ADD HEX ! ADD DEC !
976  REM **        !----------!----------!---------!---------!
980   ON W5 GOTO     1012     ,  1032    ,  1056   ,   1076  : REM BRANCH
984  REM **        !----------!----------!---------!---------!
988  REM **
992                R8$ = "00"                              : REM SET ERR MODE
996                R7$ = "Subroutine 103"                  : REM SET ERR SOURCE
1000 GOSUB ' 000                                           : REM ERR PROCESSING
1004 GOTO          1136                                    : REM BRANCH TO EXIT
1008 REM **
1012               W5 = W1*10000 + W2*100 + W3             : REM PACK PAG ADRS
1016               BIN(STR(W1$,1,1))= INT(W5/256)          : REM CONV FRST BYTE
1020               BIN(STR(W1$,2,1))=W5-INT(W5/256)*256:   REM CONV SCND BYTE
1024 GOTO          1136                                    : REM BRANCH TO EXIT
1028 REM **
1032               W5 = FNW(VAL(W1$))                      : REM CMPUT DEC VAL
1036               W1 = INT(W5/10000)                      : REM CMPUT PAGE #
1040               W2 = INT((W5-(W1*10000))/100)           : REM CMPUT ROW #
1044               W3 = W5 - (W1*10000) - (W2*100)         : REM CMPUT COL #
1048 GOTO          1136                                    : REM BRANCH TO EXIT
1052 REM **
1056 GOSUB   1032                                          : REM CONV TO DEC
1060 GOSUB   1076                                          : REM ADD TO DEC
1064 GOSUB   1012                                          : REM CONV TO HEX
1068 GOTO    1136                                          : REM BRANCH TO EXIT
1072 REM **
1076 IF            W3 + W4 > Q0(7)          THEN 1088      : REM NO SPACE ON ROW
1080               W3 = W3 + W4                            : REM INCR COL INDX
1024 GOTO          1136                                    : REM BRANCH TO EXIT
1088 IF            W2 + 1  > Q0(6)          THEN 1104      : REM NO SPACE ON FICHE
1092               W2 = W2 + 1                             : REM INCR ROW INDX
1096               W3 = W4 + W3 - Q0(7)                    : REM ADJUST COL INDX
1100 GOTO          1136                                    : REM BRANCH TO EXIT
1104 IF            W1 + 1  > Q0(5)          THEN 1124      : REM NO SPACE ON DISK
1108               W1 = W1 + 1                             : REM INCR PAGE INDX
1112               W2 = 1                                  : REM SET ROW INDX
1116               W3 = W4 + W3 - Q0(7)                    : REM ADJUST COL INDX
1120 GOTO          1136                                    : REM BRANCH TO EXIT
1124               R8$ = "00"                              : REM SET ERR CODE
1128               R7$ = "Subroutine 103"                  : REM SET SOURCE
1132 GOSUB ' 000                                           : REM ERR PROCESSING
1136 RETURN                                                : REM EXIT
```

```
1140  REM **
1144  REM **
1148  REM       Subroutine > ********************************************** 1 0 4
1152  REM       Abstract   >
1156  REM         To perform Index Lookup for the desired item.
1160  REM       Variables  >
1164  REM         R1  > item id          ! Q0$() > index buffer
1168  REM         R2  > rtrn sctr flag   ! R3$    > search id
1172  REM         R8  > indx sctr indx   ! R2$() > srch results
1176  REM         R9  > scrch indx sctr !
1180  REM **********************************************
1184  DEFFN '  104  (R1,R2)                                    : REM ENTRY
1188  MAT REDIM     Q9$(64)4                                   : REM SET I/O BUFR
1192                R2 = 0                                     : REM SET SBSCTR INDX
1196                BIN(STR(R3$,1,1)) = INT(R1/256)            : REM DECDE ITEM ID
1200                BIN(STR(R3$,2,1))=R1-INT(R1/256)*256:      REM DECDE ITEM ID
1204  FOR           R8 = 1 TO 10                               : REM BEGIN LOOP R8
1208  DATA LOAD     BA T #2, (R8,R9)Q9$()                      : REM LOAD INDX SCTR
1212  MAT SEARCH    Q9$(),=STR(R3$,1,2) TO R2$() STEP 4:       REM SRCH INDX
1216  IF            R2$(1) = HEX(0000)        THEN 1228 :      REM ID NOT FOUND
1220                R2 = INT(FNR(VAL(R2$(1)))/4) + 1           : REM CMPUT SBSCTR INDX
1224                R8 = 10                                    : REM SET END OF LOOP
1228  NEXT    R8                                               : REM CONTINUE LOOP R8
1232  RETURN                                                   : REM EXIT
```

AUTHOR System: Phase 3 - ATH1/III - # MAP

```
 0100  - 0840
 0420  - 0416
 0448  - 0436
 0516  - 0524
 0532  - 0516 0540
 0544  - 0532
 0560  - 0508
 0636  - 0628
 0660  - 0648
 0724  - 0716
 0740  - 0728
 0744  - 0736
 0756  - 0712
 0844  - 0836
 0852  - 0844
 0860  - 0852
 0868  - 0860
 0876  - 0868
 0884  - 0876
 1012  - 0980 1064
 1032  - 0980 1056
 1056  - 0980
 1076  - 0980 1060
 1088  - 1076
 1104  - 1088
 1124  - 1104
 1136  - 1004 1024 1048 1068 1084 1100 1120
 1228  - 1216
(1500) - 0848
(4000) - 0856
(6000) - 0864
(7000) - 0872
(9000) - 0880
```

```
B0     - 0232
B1     - 0232
B1$    - 0232
B1$(   - 0232
B2     - 0232
B3     - 0232
B4     - 0232
B8     - 0232
B9     - 0232
D1     - 0236
D2     - 0236
D3     - 0236
D4     - 0236
D4$    - 0236
D5     - 0236
D5$    - 0236
D6$    - 0240
D7$    - 0240
D8$    - 0240
D9$    - 0240
E1$    - 0268 0416 0428
K1     - 0244 0644 0652 0708 0732 0756
K2     - 0244 0708 0720 0744 0760
K3     - 0244 0708 0712 0724
K4     - 0244 0708 0716 0720 0748
K5     - 0724 0728 0752
K5$(   - 0244 0748
L1$    - 0248 0804 0824
L2$    - 0248 0804 0832 0840 0848 0856 0864 0872 0880
L3$    - 0248 0804 0836 0844 0852 0860 0868 0876
Q0     - 0280
Q0(    - 0284 1076 1088 1096 1104 1116
Q0$(   - 0284
Q1     - 0280
Q1$(   - 0284
Q2     - 0280
Q2$(   - 0284 0640
Q3     - 0280
Q4     - 0280
Q4$(   - 0288
Q5     - 0280
Q5$(   - 0288
Q6     - 0280
Q6$(   - 0288
Q7     - 0280
Q7$(   - 0292
Q8     - 0280
Q9     - 0280
Q9$(   - 0296 1188 1208 1212
R      - 0312
R0     - 0260
R0$    - 0264
R0$(   - 0272 0516 0532
R1     - 0260 1184 1196 1200
R1$    - 0264
R1$(   - 0296
R2     - 0260 1184 1192 1220
R2$    - 0264
```

```
R2$(   - 0272 0312 1212 1216 1220
R3     - 0260
R3$    - 0264 1196 1200 1212
R3$(   - 0272
R4     - 0260
R4$    - 0264
R4$(   - 0224
R5     - 0260
R5$    - 0264
R5$(   - 0224
R6     - 0260
R6$    - 0268
R6$(   - 0224 0304 0732 0740 0756
R7     - 0260
R7$    - 0268 0416 0432 0888 0996 1128
R7$(   - 0224 0308 0744 0760
R8     - 0260 1204 1208 1224 1228
R8$    - 0268 0436 0884 0992 1124
R8$(   - 0228 0464 0520 0536 0656 0732 0740 0744 0748 0756 0760
R9     - 0260 1208
R9$    - 0268
R9$(   - 0228
S1     - 0276 0624 0636 0644
S2     - 0276 0624 0636 0652
S3     - 0276 0624 0628 0632 0640 0656
S4     - 0276
S4$(   - 0276 0624 0632 0640 0656
S5     - 0276 0624 0648
S6     - 0636 0640
S8$(   - 0276
S9$(   - 0276
U0     - 0256
U1     - 0256
U1$(   - 0256
W      - 0316
W1     - 0252 0964 1012 1036 1040 1044 1104 1108
W1$    - 0252 0316 0964 1016 1020 1032
W2     - 0252 0964 1012 1040 1044 1088 1092 1112
W3     - 0252 0964 1012 1044 1076 1080 1096 1116
W4     - 0252 0964 1076 1080 1096 1116
W5     - 0252 0964 0980 1012 1016 1020 1032 1036 1040 1044
W6     - 0252
```

```
0412 DEFFN' 0
       - 0892 1000 1132
0336 DEFFN' 1
0344 DEFFN' 2
0352 DEFFN' 3
0360 DEFFN' 4
0368 DEFFN' 5
0500 DEFFN' 91
0624 DEFFN' 100
0708 DEFFN' 101
       - 0420 0440 0448 0504 0544 0652 0820 0828
0804 DEFFN' 102
       - 0340 0348 0356 0364 0372
0964 DEFFN' 103
1184 DEFFN' 104
```

5.2.2 ATH2/III

This section of the 'AUTHOR/3' program contains subroutines 1Ø5 to 1Ø9.

The remainder of this section presents the 'ATH2/III' program listing and reference maps.

```
1500  %## Program > AUTHOR/3  (ATH2/III)  Version > 2.0 2200 T/VP  780321
1504  REM       Subroutine > ********************************************  105
1508  REM       Function   >
1512  REM         Load storage bufer from disk
1516  REM       Variables  >
1520  REM         R0$    > begning sctr adrs! R0 > bufr indx
1524  REM         R3$    > scrtch indx adrs ! R2 > indx item flg
1528  REM         Q0$()  > transfer buffer  ! Q6() > bufr flag
1532  REM         R1$()  > storage buffer   ! R2$()> srch rsults
1536  REM         R1     > item id          !
1540  REM
1544  REM       ****************************************
1548  DEFFN ' 105   (R0$,R0,R1)                                  : REM ENTRY
1552  MAT REDIM     Q9$(64)4                                     : REM SET I/O BUFR
1556  IF            Q0$(1) <>   "1"            THEN 1576         : REM LOAD AT BEGINING
1560                STR(Q9$(1),3,2) = R0$                        : REM SET BEG SCTR ADRS
1564                R2  =  1                                     : REM SET SBSCTR INDX
1568                Q0$(1) =  "0"                                : REM RESET FLAG
1572  GOTO          1608                                         : REM BRNCH TO LOAD
1576  GOSUB ' 104   (R1,R2)                                      : REM LOCATE ITEM INDX
1580  IF            R2    <> 0                 THEN 1608         : REM ITEM FOUND
1584  GOSUB ' 101   (16,1,1,0)                                   : REM POSITION CUSUR
1588  PRINT         "  ITEM DOES NOT EXIST"                      : REM PRNT ERR MSG
1592                R8$ = "99"                                   : REM SET ERR CODE
1596                R7$ = "Subroutine 105"                       : REM SET ERR SOURCE
1600  GOSUB ' 000                                                : REM ERR PROCESSING
1604  GOTO          1696                                         : REM BRNCH TO EXIT
1608                R0$   =  STR(Q9$(R2),3,2)                    : REM STOR BEG ADRS
1612                R0    =  1                                   : REM SET BUFR INDX
1616  INIT          (20) R1$()                                   : REM CLR BUFR
1620  DATA LOAD     BA T #2, (R0$,R3$)Q9$()                      : REM LOAD DATA
1624  MAT COPY      Q9$()<1,252> TO R1$()<R0,252>                : REM LOAD BUFFR
1628  IF            Q9$(64) <> HEX(FFFFFFFF)   THEN 1676         : REM NOT END OF BUFR
1632  MAT SEARCH    R1$(), = R5$(2) TO R2$()                     : REM SRCH FOR EOI
1636  IF            R2$(1) <> HEX(0000)        THEN 1664         : REM EOI FOUND
1640  GOSUB ' 101   (16,1,1,0)                                   : REM POSITION CUSUR
1644  PRINT         " DATA IMMPROPERLY STORED"                   : REM PRNT ERR MSG
1648                R8$ = "99"                                   : REM SET ERR CODE
1652                R7$ = "Subroutine 105"                       : REM SET ERR SOURCE
1656  GOSUB ' 000                                                : REM ERR PROCESSING
1660  GOTO          1696                                         : REM BRNCH TO EXIT
1664                R0 = FNR(VAL(R2$(1)))                        : REM CMPUTE BYTE STORED
1668  MAT COPY      R5$() <1,1> TO R1$() <R0,1>                  : REM CLR EOI BYTE
1672  GOTO          1696                                         : REM BRNCH TO EXIT
1676                R0  =  R0 + 252                              : REM INCR BUFR INDX
1680                R0$   =  STR(Q9$(64),3,2)                    : REM STOR NXT ADRS
1684  IF            R0    < 2016               THEN 1620         : REM BUFR NOT FULL
1688                Q0$(1) = "1"                                 : REM SET FLAG
1692                R0    = 2016                                 : REM SET TO BUR MAX
1696  RETURN                                                     : REM EXIT
```

```
1700 REM **
1704 REM **
1708  REM **  Subroutine  ********************************************** 1 0 6
1712  REM       Function   >
1716  REM         To verify that proper output disk
1720  REM         volume is mounted. Initial new output disks.
1724  REM       Variables  >
1728  REM         Q0$()  > index buffer  ! R9 > verify mode
1732  REM         Q1$() > out indx bufr  ! R8 > out indx pos
1736  REM         R0$() > input bufr     ! R3$> sctr adrs
1740  REM         R2$() > sctr adrs sav  ! R8$> sctr adrs msk
1744  REM         Q6$() > program flags  !
1748  REM
1752  REM ***********************************************
1756  DEFFN ' 106   (R3$,R9)                                : REM ENTRY
1760  MAT REDIM     Q9$(64)4                                : REM SET I/O BUFR
1764  DATA LOAD     BA T #3, (0,R8)Q9$()                    : REM LOAD 0 SECTOR
1768                R8$ = HEX(F000)                         : REM CREATE MASK
1772                R2$(1) = R3$                            : REM SAVE SCTR ADRS
1776  AND           (R3$,R8$)                               : REM MASK SCTR ADRD
1780  IF          STR(R3$,1,1)=STR(Q9$(13),1,1)THEN 1808: REM COMPARE VOL IDS
1784  REM **
1788  REM **        !  NEW OUTPUT DISK  !  NEXT OUTPUT DISK  !
1792  REM **        !-------------------!--------------------!
1796  ON R9 GOSUB           1852        ,       2096          : REM BRANCH
1800  REM **        !-------------------!--------------------!
1804  REM **
1808                R3$    =  R2$(1)                        : REM FETCH ADDRES
1812                R8$    =  HEX(0FFF)                     : REM CREATE MASK
1816  AND           (R3$,R8$)                               : REM MASK SCTR ADRS
1820  RETURN                                                : REM EXIT
```

```
1824 REM **
1828 REM **
1832 REM **  Subroutine  ********************************************** 1 0 6. 1
1836 REM **  Abstract >
1840 REM **    To initialize a new Phase III output Disk.
1844 REM **
1848 REM ***********************************************
1852  GOSUB ' 101 (1,1,16,0)                                  : REM ENTRY/ERASE CRT
1856  GOSUB ' 101 (3,1,0,0)                                   : REM POSITION CUSOR
1860  PRINTUSING  2040, " "                                   : REM PRINT INSTRUCTIONS
1864  GOSUB ' 101 (6,1,0,0)                                   : REM POSITION CUSOR
1868  PRINTUSING  2044, " "
1872 REM **
1876  GOSUB ' 091                                             : REM RESPONCE WAIT
1880  DATA LOAD   BA T #4, (0,R8)Q9$()                        : REM LOAD SYS SCTR
1884  IF          Q9$(1) = HEX(00000000)    THEN 1916         : REM IF ZERO DISK
1888  GOSUB ' 101 (10,1,0,0)                                  : REM POSITION CUSOR
1892  PRINTUSING  2048, " "                                   : REM PRINT ERR MSG
1896  $GIO TONE   /005 (R8$(4),R8$(9))                        : REM PAUSE TONE
1900  $GIO TONE   /005 (R8$(4),R8$(9))                        : REM PAUSE TONE
1904  GOSUB ' 101 (10,1,1,0)                                  : REM ERASE ERR MSG
1908  GOTO        1876                                        : REM BRANCH TO WAIT
1912 REM **
1916  FOR         R9 = 15 TO 0 STEP -1                        : REM BEG COPY LOOP
1920  DATA LOAD   BA T #3, (R9,R8) Q9$()                      : REM LOAD IDX SCTR
1924  DATA SAVE   BA T #4, (R9,R8) Q9$()                      : REM SAVE IDX SCTR
1928  NEXT        R9                                          : REM CONT COPY LOOP
1932 REM **
1936  ADD         (STR(Q9$(13),1,1),10)                       : REM INCR VOL ID
1940              R2$(2) = STR(Q9$(13),1,2)                   : REM SAVE VOL ID
1944  DATA SAVE   BA T #4, (0,R8) Q9$()                       : REM SAVE SYS SCTR
1948  GOSUB ' 101 (6,1,1,0)                                   : REM ERASE SCREEN
1952  PRINTUSING  2052, " "                                   : REM PRINT INSTRUCTNS
1956 REM **
1960  GOSUB ' 091                                             : REM RESPONCE WAIT
1964  DATA LOAD   BA T #3, (0,R8) Q9$()                       : REM LOAD SYS SCTR
1968  IF          STR(Q9$(13),1,2) = R2$(2) THEN 2000         : REM ID'S MATCH
1972  GOSUB ' 101 (10,1,0,0)                                  : REM POSITION CUSOR
1976  PRINTUSING  2056, " "                                   : REM PRINT ERR MSG
1980  $GIO TONE   /005, (R8$(4),R8$(9))                       : REM PAUSE TONE
1984  $GIO TONE   /005, (R8$(4),R8$(9))                       : REM PAUSE TONE
1988  GOSUB ' 101 (10,1,1,0)                                  : REM ERASE ERR MSG
1992  GOTO        1852                                        : REM BRANCH TO START
1996 REM **
2000  PRINT       HEX(03)                                     : REM CLR CONSOLE DEV
2004              Q0$(2) = "1"                                : REM SET INDX RESET FLG
2008              R8$ = HEX(5102)                             : REM SET PACK INSTRUCTN
2012  FOR         R9  = 1 TO 16                               : REM BEG OUTPUT LOOP
2016  $PACK       (F=R8$)STR(R8$(9),7,2) FROM R9              : REM PACK LINE INDX
2020  $GIO /005   (R8$(6),R8$(9))Q2$()<(R9-1)*64+1,64>       : REM OUTPUT PAG BUFR
2024  NEXT        R9                                          : REM CONT OUTPUT LOOP
```

```
2028 REM **
2032 REM **                                                  OUTPUT IMAGE FORMATS
2036 REM **
2040  %  #    ******  OUTPUT  DISK  OVERFLOW  ******
2044  %  # Mount a FORMATTED Disk into SECONDARY Output Drive...
2048  %  #  **  DISK IN FORMAT DRIVE NOT FORMATTED  **
2052  %  # Replace Output Disk with New Output Disk.
2056  %  #       **  NEW DISK DOES NOT VERIFY  **
2060 REM **
2064  RETURN                                                : REM EXIT
```

```
2204 REM **
2208 REM **
2212  REM       Subroutine > ********************************************  1 0 7
2216  REM       Function   >
2220  REM         To create a block of text from storage
2224  REM         to Page buffer.
2228  REM       Variables  >
2232  REM        R1$() > storage bufer    ! R3 > begin line
2236  REM        R2$() > search buffer    ! R4 > dflt left mrgn
2240  REM        R5$() > search id's      ! R5 > dflt right mrgn
2244  REM        Q6()  > flags            ! R6 > buffer pointer
2248  REM        Q2$() > page buffer      ! R7 > length of word
2252  REM        R0$   > begin sctr adrs ! R1 > item id
2256  REM        I0    > spaces to indent!
2260  REM
2264  REM ********************************************
2268  DEFFN ' 107  (R3,R4,R5,R0$,I0)                          : REM ENTRY
2272               R6 = 1                                     : REM INIT BUFR INDX
2276               R7 = 0                                     : REM INIT WORD LENGTH
2280               Q1 = R3                                    : REM INIT LINE INDX
2284               Q2 = R4                                    : REM INIT COL INDX
2288  MAT SEARCH   R1$()<R6,2016-R6+1>,<>R5$(1)TO R2$():      REM SRCH FOR CHAR
2292  IF           R2$(1) <> HEX(0000)        THEN 2312 :     REM IF BUFR NOT EMPTY
2296  IF           Q0$(1) <> "1"              THEN 2360 :     REM BLOCK DONE
2300  GOSUB ' 105  (R0$,R0,R1)                                : REM LOAD REST OF ITEM
2304               R6    = 1                                  : REM RESET BUFR INDX
2308  GOTO         2288                                       : REM BRNCH TO SRCH
2312               R6 = FNR(VAL(R2$(1))) + R6 - 1             : REM CMPUT BUFR POS
2316  MAT SEARCH   R1$()<R6,2016-R6+1>,=R5$(1) TO R2$():      REM SRCH FOR WORD END
2320               R7 = FNR(VAL(R2$(1)))                      : REM WORD LENGTH
2324  MAT COPY     R1$()<R6,R7> TO R3$()<1,R7>                : REM EXTRACT WORD
2328  MAT COPY     R3$()<1,1>   TO R0$()<1,1>                 : REM EXTRACT 1ST BYTE
2332  IF           R0$(1) <> R5$(6)           THEN 2348 :     REM NOT KEYWORD
2336  GOSUB ' 108  (R6,R7,R3$(1),I0)                          : REM PROCESS KEYWORD
2340               R6 = R6 + R7 - 1                           : REM INCR BUFR INDX
2344  GOTO         2288                                       : REM BRNCH TO WRD SRCH
2348  GOSUB ' 109  (Q1,R4,R5,R7,R3$(1),I0)                    : REM LOAD WORD ONTO PAG
2352               R6 = R6 + R7 - 1                           : REM INCR BUFR INDX
2356  GOTO         2288                                       : REM RRNC TO WRD SRCH
2360  RETURN                                                  : REM EXIT
```

```
2364  REM **
2368  REM **
2372  REM **  Subroutine  ********************************************** 1 0 8
2376  REM **  Abstract >
2380  REM **    To execute a formating command by positioning
2384  REM **    the buffer pointer.
2388  REM **  Variables >
2392  REM **    R6 > begining bufr pos ! R7 > length of word
2396  REM **    R3$() > word           ! R8 > command indx
2400  REM **    R5 > new right margin  ! R4 > new left margin
2404  REM **    Q6$() > cmmnd indx tbl ! IO > spaces to indent
2408  REM **
2412  REM ***********************************************
2416  DEFFN ' 108  (R6,R7,R3$(1),IO)                          : REM ENTRY
2420  MAT SEARCH   Q6$(),=STR(R3$(1),2,2)TO R2$()STEP 2:      REM SRCH KEYWRD TBL
2424  IF           R2$(1) = HEX(0000)          THEN 2460 :     REM INVALID KEYWORD
2428               R8 = INT(FNR(VAL(R2$(1)))/2)+1         :    REM COMPUTE INDX
2432  REM **
2436  REM **       ! TEXT  ! WORD ! LINE ! CAPS ! TABL !
2440  REM **       !-------!------!------!------!------!
2444  ON R8 GOSUB    2492  , 2568 , 2644 , 2688 , 2732 ,
                     2916  , 2956 , 3012 , 3076 , 2460 :     REM BRANCH
2448  REM **       !-------!------!------!------!------!
2452  REM **       ! SBTITL! PAGE ! LINE ! SPACE! NULL !
2456  REM **
2460  RETURN                                                 : REM EXIT
```

```
2464  REM **
2468  REM **
2472  REM **  Subprogram  ********************************************  1 0 8 .1
2476  REM **
2480  REM **  Abstract >  To execute the '<text>' command.
2484  REM **
2488  REM ************************************************
2492                  R6 = R6 + R7 - 1                              : REM ENTRY
2496  MAT SEARCH      R1$()<R6,2016-R6+1>,=R5$(6)TO R2$()           : REM SRCH FOR NXT KEY
2500  IF              R2$(1) = HEX(0000)          THEN 2536         : REM KEY NOT FOUND
2504                  R6 = FNR(VAL(R2$(1))) + R6 - 1                : REM COMPUTE POS
2508                  R7 = 8                                        : REM SET WORD LEN
2512                  R8$ = HEX(032C)                               : REM SET UNPACK INST.
2516  $UNPACK         (D=R8$) R1$()<R6+1,R7-3> TO R4$,R5$           : REM UNPACK PRAMS
2520  IF              NUM(R4$)-NUM(R5$) <> 0      THEN 2536         : REM INVALID NUMBERS
2524  CONVERT         R4$ TO R4                                     : REM SET LEFT MARGIN
2528  CONVERT         R5$ TO R5                                     : REM SET RIGHT MARGIN
2532                  Q2 = R4                                       : REM SET LEFT MARGIN
2536  RETURN                                                        : REM EXIT
```

```
2540  REM **
2544  REM **
2548  REM **  Subprogram ********************************************  1 0 8 .2
2552  REM **
2556  REM **  Abstract >  To execute the '<word>' command.
2560  REM **
2564  REM ************************************************
2568                R8 = R6 + 6                                  : REM ENTRY
2572  GOSUB         3168                                         : REM COMPUTE BUFFR LEN
2576                R9 = R8 + R9 - 1                             : REM LST VALID BUF POS
2580  MAT SEARCH    R1$()<R8,2016-R8+1>,<>R5$(1)TO R2$():        REM SRCH FOR WORD
2584  IF            R2$(1) = HEX(0000)          THEN 2612 :      REM BUFR EMPTY
2588                R8 = FNR(VAL(R2$(1))) + R8 - 1               : REM COMPUTE BUF POS
2592  IF            R8 = R9                     THEN 2612 :      REM END OF BUFFER
2596  $TRAN         (R1$()<R8,1>, R4$()<1,64>) R                 : REM CASE CONVERSION
2600  MAT SEARCH    R1$()<R8,2016-R8+1>,= R5$(1)TO R2$():        REM SRCH FOR END OF WRD
2604                R8 = FNR(VAL(R2$(1))) + R8 - 1               : REM COMPUTE BUF POS
2608  GOTO          2580                                         : REM BRNCH TO NXT WRD
2612  RETURN                                                     : REM EXIT
```

```
2616  REM **
2620  REM **
2624  REM **  Subprogram ********************************************  1 0 8 .3
2628  REM **
2632  REM **  Abstract >  To execute the '<lower>' command.
2636  REM **
2640  REM ************************************************
2644                R8 = R6 + 7                                    : REM ENTRY
2648  GOSUB         3168                                           : REM COMPUTE BUFR LEN
2652  $TRAN         (R1$()<R8,R9>, R4$()<65,64>)R                  : REM CASE CONVERSION
2656  RETURN                                                       : REM EXIT
```

```
2660  REM **
2664  REM **
2668  REM **  Subprogram ********************************************** 1 0 8 .4
2672  REM **
2676  REM **  Abstract >  To execute the '<caps>' command.
2680  REM **
2684  REM ************************************************
2688                R8 = R6 + 6                                  : REM ENTRY
2692  GOSUB         3168                                         : REM COMPUTE BUFR LEN
2696  $TRAN         (R1$()<R8,R9>, R4$()<1,64>)R                 : REM CASE CONVERSION
2700  RETURN                                                     : REM EXIT
```

```
2704  REM **
2708  REM **
2712  REM **  Subprogram ********************************************** 1 0 8 .5
2716  REM **
2720  REM **  Abstract >  To execute the '<table>' command.
2724  REM **
2728  REM **********************************************
2732                  R6 = R6 + 7                                    : REM ENTRY
2736  MAT SEARCH      R1$()<R6,2016-R6+1>,= R5$(6)TO R2$():  REM SRCH FOR KEYWRD
2740  IF              R2$(1) = HEX(0000)          THEN 2884 : REM KEYWRD NOT FOUND
2744                  R6 = FNR(VAL(R2$(1))) + R6 - 1         : REM COMPUTE BUFR POS
2748                  R7 = 8                                 : REM SET WRD LENGTH
2752                  R8$ = HEX(032C)                        : REM SET UNPACK INST.
2756  $UNPACK         (D=R8$) R1$()<R6+1,R7-3> TO R4$,R5$ : REM UPACK PARAMS
2760  IF              NUM(R4$)-NUM(R5$) <> 0      THEN 2884 : REM INVALID NUM FMT
2764  CONVERT         R4$ TO R4                              : REM SET LEFT MARGIN
2768  CONVERT         R5$ TO R5                              : REM SET LINES IN TBL
2772                  R6 = R6 + 7
2776                  R7$= " <$>"
2780  IF              R5 > 14                     THEN 2884 : REM MAX LINES EXCLEDED
2784  IF              R5 <= 16-(Q1+1)             THEN 2808 : REM SPACE AVAIABLE
2788  GOSUB ' 100(16,41,23,"To continue Go to :nxt0",Q0(12))
2792  GOSUB ' 110   (D1,D2,D3,D4$,D5)                        : REM STOR PAG ON DISK
2796  GOSUB ' 101   (2,1,15,0)                               : REM CLR CRT SCREEN
2800  $TRAN         (Q2$()<65,960>, R4$()<177,4>)00 R        : REM CLR PAG BUFR
2804                  Q1 =   1                               : REM SET BUFR LIN INDX
2808                  Q1 = Q1 + 1                            : REM INCR BUFR LIN IDX
2812                  Q2 = R4                                : REM RESET LEFT MARGIN
2816  FOR             R8 = 1 TO R5                           : REM PROCES ALL LINES
2820  MAT  SEARCH     R1$()<R6,2016-R6+1>,=  R7$ TO R2$()    : REM SRCH STORAGE BUFR
2824                  R6 = FNR(VAL(R2$(1))) + R6 + 3         : REM COMPUTE POSITION
2828                  R9 = 64                                : REM SET TO MAX LENGTH
2832  IF              2016-R6+1 > 63              THEN 2840 : REM NOT END OF BUFFR
2836                  R9 = 2016-R6+1                         : REM SET TO MAX LEGNTH
2840  MAT  COPY       R1$()<R6,R9> TO S4$()<1,64>            : REM LOAD LIN TO BUFR
2844  MAT  SEARCH     S4$(), =  R5$(6) TO R2$()              : REM SRCH FOR END
2848                  R7 = FNR(VAL(R2$(1)))                  : REM COMPUTE POSITION
2852  IF              R7 <> 0                     THEN 2860 : REM EOI FOUND
2856                  R7 = 64                                : REM SET TO MAX LENGTH
2860  MAT  COPY       R5$()<1,1> TO S4$()<R7,64-R7+1>        : REM CLR END OF BUFR
2864  GOSUB ' 100   (Q1,Q2,64-R4+1,S4$(1),Q0(12))            : REM LOAD PAG BUFR
2868                  Q1 = Q1 + 1                            : REM INCR BUFR LIN IDX
2872                  R6 = R6 + R7 - 2                       : REM INCR STOR BUF IDX
2876  NEXT            R8                                     : REM CONT PROCESSING
2880                  R7 = 5                                 : REM SET LAST KEY LEN
2884  RETURN                                                 : REM EXIT
```

```
2888  REM **
2892  REM **
2896  REM **  Subprogram ********************************************  1 0 8 .6
2900  REM **
2904  REM **  Abstract >  To execute the '<subtitle>' command.
2908  REM **
2912  REM ***********************************************
2916                R4 = 6                                          : REM ENTRY
2920                R5 = 22
2924  RETURN                                                        : REM EXIT
```

```
2928  REM **
2932  REM **
2936  REM **  Subprogram ********************************************  1 0 8 .7
2940  REM **
2944  REM **  Abstract >  To execute the '<page>' command.
2948  REM **
2952  REM ***********************************************
2956                Q1 = 2                                    : REM ENTRY
2960                Q2 = R4 + IO                              : REM SET LEFT MARGIN
2964  GOSUB ' 100(16,41,23,"To continue Co to :nxtO",QO(12))
2968  GOSUB ' 110  (D1,D2,D3,D4$,D5)                          : REM STOR PAG ON DISK
2972  GOSUB ' 101 (2,1,15,O)
2976  $TRAN       (Q2$()<65,960>, R4$()<177,4>)OO R           : REM CLR PAG BUFR
2980  RETURN                                                  : REM EXIT
```

```
2984  REM **
2988  REM **
2992  REM **  Subprogram ********************************************  1 0 8 .8
2996  REM **
3000  REM **  Abstract >  To execute the '<line>' command.
3004  REM **
3008  REM ***********************************************
3012              Q1 = Q1 + 1                                  : REM ENTRY
3016              Q2 = R4                                      : REM RESET LEFT MARGIN
3020  IF          Q1 < 16                    THEN 3044         : REM NOT LAST LINE
3024  GOSUB ' 100 (16,41,23,"To continue Go to :nxt0",Q0(12))
3028  GOSUB ' 110 (D1,D2,D3,D4$,D5)                            : REM STOR PAG ON DISK
3032  GOSUB ' 101 (2,1,15,0)                                   : REM CLR CRT SCREEN
3036  $TRAN       (Q2$()<65,960>, R4$()<177,4>)00 R            : REM CLR PAG BUFR
3040              Q1 = 2                                       : REM RESET LINE INDX
3044  RETURN                                                   : REM EXIT
```

```
3048 REM **
3052 REM **
3056 REM **  Subroutine  ********************************************** 1 0 8 .9
3060 REM **
3064 REM **  Abstract >  To execute the '<space>' command.
3068 REM **
3072 REM ************************************************
3076                 R6 = R6 + R7 - 1                          : REM ENTRY
3080  MAT SEARCH     R1$()<R6,2016-R6+1>,=R5$(6) TO R2$():     REM SRCH RST OF CMND
3084  IF             R2$(1) = HEX(0000)          THEN 3124 :   REM INVALID CMND SEQ
3088                 R6 = FNR(VAL(R2$(1))) + R6 - 1        :   REM INCR POSITION
3092                 R7 = 5                                :   REM STOR RESULT
3096                 R8$ = HEX(A002)                       :   REM SET UNPAK FORMAT
3100  $UNPACK        (F=R8$) R1$()<R6+1,R7-3> TO R4$       :   REM FETCH TAB
3104  IF             NUM(R4$) <> 2               THEN 3124 :   REM INVALID FORMAT
3108  CONVERT        R4$ TO R9                             :   REM CMPUT TOTAL MOVE
3112                 Q2 = Q2 + R9                          :   REM UPDATE INDEX
3116  IF             Q2 < R5                     THEN 3124 :   REM WITHEN MARGIN
3120  GOSUB 3012                                           :   REM INCR LINE INDX
3124  RETURN                                               :   REM EXIT
```

```
3128  REM **
3132  REM **
3136  REM **  Subprogram  ********************************************* 1 0 8 .10
3140  REM **  Abstract > To compute the length of the buffer
3144  REM **             to which modifications are to be
3148  REM **             performed on.
3152  REM **  Note >    special computations...
3156  REM **              > R8+R9-2    last valid char position
3160  REM **              > R8+R9-1    position of nxt command
3164  REM ***********************************************
3168  MAT SEARCH   R1$()<R8,2016-R8+1>,= R5$(6)TO R2$(): REM ENTRY
3172  IF           R2$(1) <> HEX(0000)       THEN 3184 : REM KEYWRD FOUND
3176               R9 = 2016 - R8 + 1                  : REM SET TO BUFR MAX
3180  GOTO         3188                                : REM BRNCH TO EXIT
3184               R9 = FNR(VAL(R2$(1))) - 1           : REM COMPUTE BUFR LEN
3188  RETURN                                           : REM EXIT
```

```
3192  REM **
3196  REM **
3200  REM **  Subroutine  ********************************************** 1 0 9
3204  REM **  Abstract >
3208  REM **    To format the input word withen margins and
3212  REM **    to control page formatting and storage control.
3216  REM **  Variables >
3220  REM **    Q1 > current row indx ! Q2 > current col indx
3224  REM **    R4 > left margin      ! R5 > right margin
3228  REM **    I0 > spaces to indent !
3232  REM **
3236  REM ***********************************************
3240  DEFFN ' 109   (Q1,R4,R5,R7,R3$(1),I0)                    : REM ENTRY
3244  IF            Q2+R7-1 <= R5              THEN 3280      : REM WORD FITS ON LINE
3248                Q1 = Q1 + 1                               : REM INCR LINE INDX
3252                Q2 = R4 + I0                              : REM SET LEFT MARIGN
3256  IF            Q1 < 16                    THEN 3280      : REM NOT LAST LINE
3260                Q1 = 2                                    : REM SET TO T-O-P
3264  GOSUB ' 100(16,41,23,"To continue Go to :nxt0",Q0(12))
3268  GOSUB ' 110   (D1,D2,D3,D4$,D5)                         : REM SAVE CURNT PAG
3272  GOSUB ' 101   (2,1,15,0)                                : REM ERASE CRT TEXT
3276  $TRAN         (Q2$()<65,960>, R4$()<177,4>)00 R         : REM ERAS BUFFER
3280  GOSUB ' 100   (Q1,Q2,R7,R3$(1),Q0(12))                  : REM WRITE WORD
3284                Q2 = Q2 + R7                              : REM INCR COL INDX
3288  RETURN                                                  : REM EXIT
```

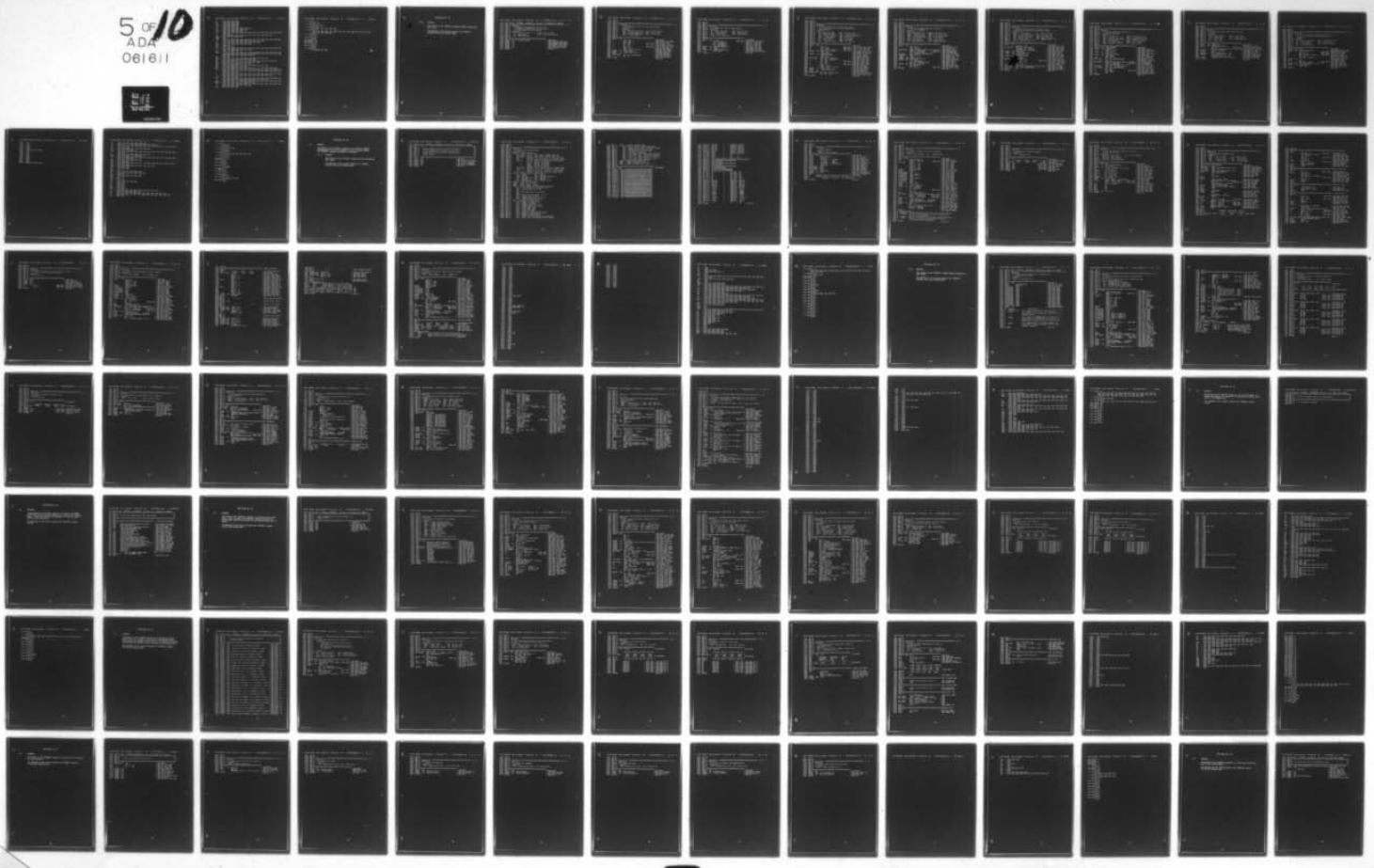
AUTHOR System: Phase 3 - ATH2/III - # MAP

1576 - 1556
1608 - 1572 1580
1620 - 1684
1664 - 1636
1676 - 1628
1696 - 1604 1660 1672
1808 - 1780
1852 - 1796 1992
1876 - 1908
1916 - 1884
2000 - 1968
2040 - 1860
2044 - 1868
2048 - 1892
2052 - 1952
2056 - 1976
2096 - 1796
2124 - 2168
2148 - 2140
2184 - 2108
2188 - 2116
2192 - 2152
2200 - 2144
2288 - 2308 2344 2356
2312 - 2292
2348 - 2332
2360 - 2296
2460 - 2424 2444
2492 - 2444
2536 - 2500 2520
2568 - 2444
2580 - 2608
2612 - 2584 2592
2644 - 2444
2688 - 2444
2732 - 2444
2808 - 2784
2840 - 2832
2860 - 2852
2884 - 2740 2760 2780
2916 - 2444
2956 - 2444
3012 - 2444 3120
3044 - 3020
3076 - 2444
3124 - 3084 3104 3116
3168 - 2572 2648 2692
3184 - 3172
3188 - 3180
3280 - 3244 3256

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U)
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

5 OF 10
ADA
061611



AUTHOR System: Phase 3 - ATH2/III - V MAP

```
D1     - 2792 2968 3028 3268
D2     - 2792 2968 3028 3268
D3     - 2792 2968 3028 3268
D4$    - 2792 2968 3028 3268
D5     - 2792 2968 3028 3268
I0     - 2268 2336 2348 2416 2960 3240 3252
G0(    - 2788 2864 2964 3024 3264 3280
G0$(   - 1556 1568 1688 2004 2296
Q1     - 2280 2348 2784 2804 2808 2864 2868 2956 3012 3020 3040 3240 3248 3256
         3260 3280
Q2     - 2284 2532 2812 2864 2960 3016 3112 3116 3244 3252 3280 3284
Q2$(   - 2020 2800 2976 3036 3276
Q6$(   - 2420
Q9$(   - 1552 1560 1608 1620 1624 1628 1680 1760 1764 1780 1880 1884 1920 1924
         1936 1940 1944 1964 1968 2128 2132 2140 2144 2152
R0     - 1548 1612 1624 1664 1668 1676 1684 1692 2300
R0$    - 1548 1560 1608 1620 1680 2268 2300
R0$(   - 2328 2332
R1     - 1548 1576 2300
R1$(   - 1616 1624 1632 1668 2288 2316 2324 2496 2516 2580 2596 2600 2652 2696
         2736 2756 2820 2840 3080 3100 3168
R2     - 1564 1576 1580 1608
R2$(   - 1632 1636 1664 1772 1808 1940 1968 2288 2292 2312 2316 2320 2420 2424
         2428 2496 2500 2504 2580 2584 2588 2600 2604 2736 2740 2744 2820 2824
         2844 2848 3080 3084 3088 3168 3172 3184
R3     - 2268 2280
R3$    - 1620 1756 1772 1776 1780 1808 1816 2096 2108 2140 2152
R3$(   - 2324 2328 2336 2348 2416 2420 3240 3280
R4     - 2268 2284 2348 2524 2532 2764 2812 2864 2916 2960 3016 3240 3252
R4$    - 2516 2520 2524 2756 2760 2764 3100 3104 3108
R4$(   - 2596 2652 2696 2800 2976 3036 3276
R5     - 2268 2348 2528 2768 2780 2784 2816 2920 3116 3240 3244
R5$    - 2516 2520 2528 2756 2760 2768
R5$(   - 1632 1668 2288 2316 2332 2496 2580 2600 2736 2844 2860 3080 3168
R6     - 2272 2288 2304 2312 2316 2324 2336 2340 2352 2416 2492 2496 2504 2516
         2568 2644 2688 2732 2736 2744 2756 2772 2820 2824 2832 2836 2840 2872
         3076 3080 3088 3100
R7     - 2276 2320 2324 2336 2340 2348 2352 2416 2492 2508 2516 2748 2756 2848
         2852 2856 2860 2872 2880 3076 3092 3100 3240 3244 3280 3284
R7$    - 1596 1652 2776 2820
R8     - 1764 1880 1920 1924 1944 1964 2128 2428 2444 2568 2576 2580 2588 2592
         2596 2600 2604 2644 2652 2688 2696 2816 2876 3168 3176
R8$    - 1592 1648 1768 1776 1812 1816 2008 2016 2512 2516 2752 2756 3096 3100
R8$(   - 1896 1900 1980 1984 2016 2020 2156 2160
R9     - 1756 1796 1916 1920 1924 1928 2012 2016 2020 2024 2576 2592 2652 2696
         2828 2836 2840 3108 3112 3176 3184
S4$(   - 2840 2844 2860 2864
```

AUTHOR System: Phase 3 - ATH2/III - ' MAP

```
???? DEFFN' 0
       - 1600 1656
???? DEFFN' 91
       - 1876 1960 2124
???? DEFFN' 100
       - 2788 2864 2964 3024 3264 3280
???? DEFFN' 101
       - 1584 1640 1852 1856 1864 1888 1904 1948 1972 1988 2100 2104 2112 2148
         2164 2796 2972 3032 3272
???? DEFFN' 104
       - 1576
1548 DEFFN' 105
       - 2300
1756 DEFFN' 106
2268 DEFFN' 107
2416 DEFFN' 108
       - 2336
3240 DEFFN' 109
       - 2348
???? DEFFN' 110
       - 2792 2968 3028 3268
```

5.2.3 ATH3/III

This section of the 'AUTHOR/3' program contains subroutines 11Ø to 119.

The remainder of this section presents the 'ATH3/III' program listing and reference maps.

```
3400  %## Program > AUTHOR/3  (ATH3/III)  Version > 2.0 2200 T/VP  780321
3402  REM ** Subroutine ************************************************** 1 1 0
3404  REM **   Abstract >
3406  REM **     To Output the page buffer to the disk.
3408  REM **     Assigning a page address and evaluating
3410  REM **     any references to the next page.
3412  REM **  Variables >
3414  REM **    D1 > indx sctr adrs        ! D2 > sub sctr indx
3416  REM **    D3 > page flag             ! D4$ > previous pag indx
3418  REM **    D5 > indexing mode         !
3420  REM **
3422  REM ********************************************
3424  DEFFN ' 110  (D1,D2,D3,D4$,D5)                    : REM ENTRY
3426  GOSUB ' 111                                       : REM FTCH NXT INDX SLOT
3428  GOSUB ' 112                                       : REM COMPUTE PAGE INDEX
3430  GOSUB ' 113                                       : REM RESERVE INDX SPACE
3432  GOSUB ' 114                                       : REM CHECK FOR PAG REF
3434  GOSUB ' 115                                       : REM STORE PAG ON DSK
3436  RETURN                                            : REM EXIT
```

```
3438  REM **
3440  REM **
3442  REM ** Subroutine ************************************************** 1 1 1
3444  REM **    Abstract >
3446  REM **      To locate next index slot and load previous
3448  REM **      page address.
3450  REM **    Variables >
3452  REM **      D2 > indx sub sctr adrs  ! D1 > indx sctr adrs
3454  REM **      D4$ > previous page adrs ! R8$ > error code
3456  REM **      D5$ > current page adrs  ! R9$ > error code
3458  REM **      Q1$() > index buffer     !
3460  REM **
3462  REM ********************************************
3464  DEFFN ' 111                                          : REM ENTRY
3466                D5$ = D4$                              : REM INITIAL PAG ADRS
3468                D2  = D2 + 2                           : REM INCRMNT INDX
3470  IF            D2 < 128                THEN 3490    : REM IF NOT AT END
3472                D1 = D1 + 1                            : REM INCRMNT SCTR ADRS
3474  IF            D1 < 16                 THEN 3486    : REM IF INDX NOT FULL
3476                R8$ = "xx"                             : REM SET ERR MODE
3478                R9$ = "Subroutine 111"                 : REM SET ERR SOURCE
3480  GOSUB ' 101   (16,1,1,40)                            : REM POSITION CUSOR
3482  PRINT         "INDEX AREA FULL!!"                    : REM ERR MESSAGE
3484  GOSUB ' 000                                          : REM ERR PROCESSING
3486  DATA LOAD     BA T #3, (D1,D2)Q1$()                  : REM LOAD NXT SCTR
3488                D2 = 1                                 : REM INITIAL INDX
3490  RETURN                                               : REM EXIT
```

```
3492  REM **
3494  REM **
3496  REM ** Subroutine ************************************************** 1 1 2
3498  REM **  Abstract >
3500  REM **    To compute the current page address.
3502  REM **  Variables >
3504  REM **    D1 > indx sctr adrs     ! D2 > sub sctr indx
3506  REM **    W1 > fiche page #       ! W2 > fiche row #
3508  REM **    W3 > fiche col #        ! W1$ > page address
3510  REM **    D5$ > currnt pag adrs  !
3512  REM **
3514  REM ********************************************
3516  DEFFN ' 112                                          : REM ENTRY
3518  IF            D1+D2 <> 2                  THEN 3530  : REM IF NOT FIRST PAG
3520                W1 = VAL(GO$(6))                       : REM SET FICHE PAGE
3522                W2 = VAL(GO$(7))                       : REM SET FICHE ROW
3524                W3 = VAL(GO$(8))                       : REM SET FICHE COL
3526  GOSUB ' 103   (W1$,W1,W2,W3,0,1)                     : REM CMPUT PAG ADRS
3528  GOTO          3532                                   : REM BRANCH TO EXIT
3530  GOSUB ' 103   (D5$,W1,W2,W3,1,3)                     : REM CMPUT PAG ADRS
3532                D5$ = W1$                              : REM STORE NEW PAG ADRS
3534  RETURN                                               : REM EXIT
```

```
3536  REM **
3538  REM **
3540  REM ** Subroutine ************************************************** 1 1 3
3542  REM **  Abstract >
3544  REM **    To reserve space for index at end of each
3546  REM **    filled fiche if a fiche index sequence has
3548  REM **    been requested.
3550  REM **  Variables >
3552  REM **    D1 > indx sctr adrs     ! D2 > sub sctr indx
3554  REM **    D5 > sequening mode     ! D4$ > previous pag adr
3556  REM **    D5$ > currnt pag adrs   ! D6$ > redefned pag adr
3558  REM **    Q1$() > index buffer    ! W1$ > pag addrs
3560  REM **    W1 > fiche page #       ! W2 > fiche row #
3562  REM **    W3 > fiche col #        ! R8$ > err mode
3564  REM **    QO() > register stack   ! R9$ > err source
3566  REM **
3568  REM ********************************************
3570  DEFFN ' 113                                          : REM ENTRY
3572  IF            D5 = 3                    THEN 3612    : REM IF SEQU INDX MODE
3574  IF            W2 <> QO(6)               THEN 3612    : REM IF NOT LAST ROW
3576  IF            W3 <> QO(7)-(QO(8)-1)     THEN 3612    : REM IF NOT LAST COL
3578  GOSUB ' 103   (W1$,W1+1,1,1,0,1)                     : REM CMPUT NEW PAG REF
3580                D6$ = W1$                              : REM STORE NEW PAG REF
3582  GOSUB ' 117   (D4$,D5$,D6$)                          : REM CHANGE PAG REF
3584                Q1$(D2) = D5$                          : REM UPDATE INDEX
3586                D5$ = D6$                              : REM REDEFN PAG ADRS
3588  DATA SAVE     BA T #3, (D1,DO)Q1$()                  : REM SAVE INDX SCTR
3590                D2 = D2 + (2*QO(8))                    : REM INCR SCTR INDX
3592  IF            D2 < 128                  THEN 3612    : REM NOT END OF SCTR
3594                D1 = D1 + 1                            : REM INCR SCTR ADRS
3596  IF            D1 < 16                   THEN 3608    : REM NOT END OF INDX
3598                R8$ = "xx"                             : REM SET ERR MODE
3600                R9$ = "Subroutine 113"                 : REM SET ERR SOURCE
3602  GOSUB ' 101   (16,1,1,40)                            : REM POSITION CUSOR
3604  PRINT         "INDEX AREA FULL!!"                    : REM ERR MESSAGE
3606  GOSUB ' 000                                          : REM ERR PROCESSING
3608  DATA LOAD     BA T #3, (D1,RO) Q1$()                 : REM LOAD NXT INDX SCTR
3610                D2 = D2 - 128                          : REM ADJUST SCTR INDX
3612  RETURN                                               : REM EXIT
```

```
3614  REM **
3616  REM **
3618  REM ** Subroutine ************************************************** 1 1 4
3620  REM **  Abstract >
3622  REM **    To search the output page for references
3624  REM **    to the "nxt" command.
3626  REM **  Variables >
3628  REM **    D6 > output bufr indx  ! D7$ > temp search id
3630  REM **    Q2$() > output bufr    ! R2$() > srch rslt buf
3632  REM **    RO$() > charactr bufr  ! RO > # of sequintl pag
3634  REM **    D5$ > current pag adrs ! W1 > fiche page #
3636  REM **    W2 > fiche row #       ! W3 > fiche col #
3638  REM **    W1$ > page address     ! D7$ > page # string
3640  REM **    S1 > output buf row    ! S2 > output buf col
3642  REM **
3644  REM *********************************************
3646  DEFFN ' 114                                          : REM ENTRY
3648                D6 = Q0(9)                             : REM OUT BUFR INDX
3650                D7$ = ":nxt"                           : REM SEARCH ID
3652  MAT SEARCH    Q2$()<D6,Q0(10)-D6+1>,=D7$ TO R2$()    : REM SEARCH BUFFER
3654  IF            R2$(1) = HEX(0000)          THEN 3678  : REM NO REFERENCE
3656                D6 = FNR(VAL(R2$(1))) + D6 - 1         : REM INCR BUF INDX
3658  MAT COPY      Q2$()<D6+4,1> TO RO$()<1,1>            : REM FETCH # OF PAGS
3660  CONVERT       RO$(1) TO RO                           : REM CONVERT TO <DEC>
3662  GOSUB ' 119   (RO)                                   : REM RESRV SPACE
3664  GOSUB ' 103   (D5$,W1,W2,W3,1,3)                     : REM CMPUT PAGE ADRS
3666  GOSUB ' 118   (W1$,D7$)                              : REM CONSTRT PAGE #
3668                S1 = INT(D6/64) + 1                    : REM CMPUT ROW #
3670                S2 = D6 - INT(D6/64)*64                : REM CMPUT COL #
3672  GOSUB ' 100   (S1,S2,5,D7$,00)                       : REM WRITE PAGE #
3674                D6 = D6 + 5                            : REM INCR BUFR INDX
3676  IF            D6 < Q0(10)               THEN 3650    : REM NOT END OF BUFR
3678  RETURN                                               : REM EXIT
```

```
3680  REM **
3682  REM **
3684  REM ** Subroutine ************************************************** 1 1 5
3686  REM **  Abstract >
3688  REM **    To update index and index pointers. Insert
3690  REM **    current page address and save page on disk.
3692  REM **  Variables >
3694  REM **    Q1$() > index buffer      ! D2 > sub sctr indx
3696  REM **    Q9$() > i/o transfer buf  ! D1 > indx sctr adrs
3698  REM **    R0 > scratch sctr adrs    ! D4$ > previous pag
3700  REM **    D5$ > current pag adrs    ! D7$ > pag # string
3702  REM **    R9 > scratch indx         ! Q4 > memory buffer indx
3704  REM **    Q4$() > pag adrs mem bufr!
3706  REM **
3708  REM ***********************************************
3710  DEFFN ' 115                                          : REM ENTRY
3712                Q1$(D2) = D5$                          : REM UPDATE INDX TBL
3714  DATA SAVE     BA T #3, (D1,R0)Q1$()                  : REM SAVE INDX SCTR
3716                STR(D4$,1,2) = Q1$(D2)                 : REM STOR PAGE ADRS
3718                STR(D4$,3,2) = Q1$(D2+1)               : REM STOR BEG SCTR
3720  GOSUB ' 118   (D5$,D7$)                              : REM CREATE PAGE #
3722  GOSUB ' 100   (1,59,5,D7$,00)                        : REM WRITE PAGE #
3724                BIN(D7$) = D3                          : REM CMPUT PAG INDX
3726  GOSUB ' 100   (1,63,1,D7$,01)                        : REM WRITE PAG INDX
3728                D3 = 32                                : REM CLR PAG INDX
3730  GOSUB ' 105   (Q1$(D2+1),1)                          : REM MOUNT OUT DISK
3732  IF            Q4 = 00                  THEN 3738     : REM FORGET PAG ADRS
3734                Q4$(Q4) = D5$                          : REM STOR IN MEM BUFR
3736                Q4 = 00                                : REM CLR MEMORY FLAG
3738  FOR           R9 = 1 TO 4                            : REM BEGIN LOOP R9
3740  MAT COPY      Q2$()<(R9-1)*256+Q0(9),256>TO Q9$()    : REM LOAD I/O BUFR
3742  DATA SAVE     BA T #3, (R3$,R3$)Q9$()                : REM SAVE I/O BUFR
3744  NEXT          R9                                     : REM CONTINUE LOOP R9
3746  RETURN                                               : REM EXIT
```

```
3748  REM **
3750  REM **
3752  REM ** Subroutine ************************************************** 1 1 7
3754  REM **  Abstract >
3756  REM **    To search a page for reference to a
3758  REM **    redefined page number.
3760  REM **  Variables >
3762  REM **    S8$() > search id          ! S9$() > subsitude id
3764  REM **    D8$ > begining sctr adrs   ! D4$ > previous pag indx
3766  REM **    D5$ > current page addrs   ! D6$ > redefined pag adrs
3768  REM **    D7$ > page number          ! R3$ > masked sctr adrs
3770  REM **    D9$ > sctr adrs register   ! G9$() > i/o buffer
3772  REM **    R2$() > search rslt bufr   ! D0 > buffer indx
3774  REM **
3776  REM ********************************************
3778  DEFFN ' 117   (D4$,D5$,D6$)                          : REM ENTRY
3780  GOSUB ' 118   (D5$,D7$)                              : REM CREATE PAC #
3782                S8$(1) = D7$                           : REM STORE PAG # SRCH
3784  GOSUB ' 118   (D6$,D7$)                              : REM CREAT PAG #
3786                S9$(1) = D7$                           : REM STORE PAC # NEW
3788  GOSUB ' 106   (STR(D4$,3,2),2)                       : REM MOUNT DISK
3790                D8$ = R3$                              : REM STOR SCTR ADRS
3792  FOR           R9 = 1 TO 4                            : REM BEGIN LOOP R9
3794  DATA LOAD     BA T #3, (D8$,D9$)G9$()                : REM LOAD 1/4 PAGE
3796                D0 = 1                                 : REM BUFFER INDX STRT
3798  MAT SEARCH    G9$()<D0,256-D0+1>,=S8$(1) TO R2$()    : REM SEARCH BUFR
3800  IF            R2$(1) = HEX(0000)          THEN 3812  : REM NO REFERENCE
3802                D0 = FNR(VAL(R2$(1))) + D0 - 1         : REM INCR BUFR INDX
3804  MAT COPY      S9$()<1,4> TO G9$()<D0,4>              : REM INSRT NEW PAC #
3806  DATA SAVE     BA T #3, (D8$,D9$)G9$()                : REM SAVE 1/4 PAGE
3808                D0 = D0 + 4                            : REM INCR BUFR INDX
3810  IF            D0 < 256                    THEN 3798  : REM NOT END OF SCTR
3812                D8$ = D9$                              : REM ASSIGN NXT ADRS
3814  NEXT          R9                                     : REM CONTINUE LOOP R9
3816  RETURN                                               : REM EXIT
```

```
3818  REM **
3820  REM **
3822  REM ** Subroutine ************************************************** 1 1 8
3824  REM **  Abstract >
3826  REM **    To construct a page number from the page
3828  REM **    address.
3830  REM **  Variables >
3832  REM **    D8$ > page address      ! D7$ > page number
3834  REM **    W1 > fiche page #       ! W2 > fiche row #
3836  REM **    W3 > fiche col #        ! D5 > formatting mode
3838  REM **    W5 > scale index        !
3840  REM **
3842  REM ********************************************
3844  DEFFN ' 118  (D8$,D7$)                                    : REM ENTRY
3846  GOSUB ' 103  (D8$,W1,W2,W3,0,2)                           : REM CONV TO <DEC>
3848  IF           D5 <> 3                          THEN 3856  : REM NOT SEQUNTL FORM
3850               W5=(W1-1)*QO(6)*QO(7)+(W2-1)*QO(7)+W3: REM SCALE TO SEQ FORM
3852  CONVERT      W5 TO D7$,(####)                             : REM CONV TO DISPLY
3854  GOTO         3864                                         : REM BRANCH TO EXIT
3856  CONVERT      W1 TO STR(D7$,1,1), (#)                      : REM CONV FICHE PAC #
3858  CONVERT      W3 TO STR(D7$,3,2), (##)                     : REM CONV FICHE COL #
3860               BIN(STR(D7$,2,1)) = W2                       : REM CONV FICHE ROW #
3862  $TRAN        (STR(D7$,2,1), R4$()<129,18>)                : REM CONV TO LETTER
3864  RETURN                                                    : REM EXIT
```

```
3866  REM **
3868  REM **
3870  REM ** Subroutine ************************************************** 1 1 9
3872  REM **  Abstract >
3874  REM **    To check if there are enough pages left on a
3876  REM **    row of fiche to insert the ordered page
3878  REM **    formats.
3880  REM **  Variables >
3882  REM **    R0 > pages in sequence  ! D5$ > current pag adrs
3884  REM **    W1 > fiche page #       ! W2 > fiche row #
3886  REM **    W3 > fiche col #        ! D5 > formatting mode
3888  REM **    Q6() > register stack   !
3890  REM **
3892  REM ************************************************
3894  DEFFN ' 119  (R0)                                      : REM ENTRY
3896  GOSUB ' 103  (D5$,W1,W2,W3,0,2)                        : REM CONV ADRS TO <DEC>
3898  IF           D5 = 3                          THEN 3920 : REM SEQUNTL FORMAT
3900  IF           W2 <> Q0(6)                     THEN 3912 : REM NOT LAST ROW
3902  IF           R0+W3 <= Q0(7)-(Q0(8)-1)        THEN 3920 : REM ENOUGH ROOM
3904               W2 = Q0(6)                               : REM SET TO LAST ROW
3906               W3 = Q0(7)-(Q0(8)-1)                     : REM SET TO INDX POS
3908  GOSUB ' 113                                           : REM RESERVE INDX
3910  GOTO         3920                                     : REM BRANCH TO EXIT
3912  IF           R0 + W3 <= Q0(7)                THEN 3920 : REM ENOUGH ROOM
3914  GOSUB ' 103  (D5$,W1,W2,W3,Q0(7)-(W3-1),3)             : REM INCR PAGE ADRS
3916  GOSUB ' 117  (D4$,D5$,W1$)                            : REM UPDAT PAGE REF
3918               D5$ = D6$                                : REM STOR PAGE ADDRS
3920  RETURN                                                : REM EXIT
```

AUTHOR System: Phase 3 - ATH3/III - # MAP

3486 - 3474
3490 - 3470
3530 - 3518
3532 - 3528
3608 - 3596
3612 - 3572 3574 3576 3592
3650 - 3676
3678 - 3654
3738 - 3732
3798 - 3810
3812 - 3800
3856 - 3848
3864 - 3854
3912 - 3900
3920 - 3898 3902 3910 3912

AUTHOR System: Phase 3 - ATH3/III - V MAP

```
D0     - 3588 3796 3798 3802 3804 3808 3810
D1     - 3424 3472 3474 3486 3518 3588 3594 3596 3608 3714
D2     - 3424 3468 3470 3486 3488 3518 3584 3590 3592 3610 3712 3716 3718 3730
D3     - 3424 3724 3728
D4$    - 3424 3466 3582 3716 3718 3778 3788 3916
D5     - 3424 3572 3848 3898
D5$    - 3466 3530 3532 3582 3584 3586 3664 3712 3720 3734 3778 3780 3896 3914
         3916 3918
D6     - 3648 3652 3656 3658 3668 3670 3674 3676
D6$    - 3580 3582 3586 3778 3784 3918
D7$    - 3650 3652 3666 3672 3720 3722 3724 3726 3780 3782 3784 3786 3844 3852
         3856 3858 3860 3862
D8$    - 3790 3794 3806 3812 3844 3846
D9$    - 3794 3806 3812
Q0(    - 3574 3576 3590 3648 3652 3676 3740 3850 3900 3902 3904 3906 3912 3914
Q0$(   - 3520 3522 3524
Q1$(   - 3486 3584 3588 3608 3712 3714 3716 3718 3730
Q2$(   - 3652 3658 3740
Q4     - 3732 3734 3736
Q4$(   - 3734
Q9$(   - 3740 3742 3794 3798 3804 3806
R0     - 3608 3660 3662 3714 3894 3902 3912
R0$(   - 3658 3660
R2$(   - 3652 3654 3656 3798 3800 3802
R3$    - 3742 3790
R4$(   - 3862
R8$    - 3476 3598
R9     - 3738 3740 3744 3792 3814
R9$    - 3478 3600
S1     - 3668 3672
S2     - 3670 3672
S8$(   - 3782 3798
S9$(   - 3786 3804
W1     - 3520 3526 3530 3578 3664 3846 3850 3856 3896 3914
W1$    - 3526 3532 3578 3580 3666 3916
W2     - 3522 3526 3530 3574 3664 3846 3850 3860 3896 3900 3904 3914
W3     - 3524 3526 3530 3576 3664 3846 3850 3858 3896 3902 3906 3912 3914
W5     - 3850 3852
```

```
???? DEFFN' 0
       - 3484 3606
???? DEFFN' 100
       - 3672 3722 3726
???? DEFFN' 101
       - 3480 3602
???? DEFFN' 103
       - 3526 3530 3578 3664 3846 3896 3914
???? DEFFN' 106
       - 3730 3788
3424 DEFFN' 110
3464 DEFFN' 111
       - 3426
3516 DEFFN' 112
       - 3428
3570 DEFFN' 113
       - 3430 3908
3646 DEFFN' 114
       - 3432
3710 DEFFN' 115
       - 3434
3778 DEFFN' 117
       - 3582 3916
3844 DEFFN' 118
       - 3666 3720 3780 3784
3894 DEFFN' 119
       - 3662
```

5.3 ATH3INIT

The purpose of the 'ATH3INIT' program is to initialize program variables and to allow the operator to select creation options. The 'ATH3INIT' program is presented in two parts.

5.3.1 ATH3INT1

This section of the 'ATH3INIT' program contains subroutines 2Ø1 to 2Ø9.

The remainder of this section presents the 'ATH3INT1' program listing and reference maps.

```
4000  %## Program > AUTHOR/3  (ATH3INT1)  Version 2.0  2200 T/VP   780321
4004  REM **  Abstract  ************************************************
4008  REM **                                                           **
4012  REM **    Overlay ATH3INIT will initialize all program           **
4016  REM **    variable tables including the initialing of            **
4020  REM **    of the Symbol Table.  ATH3INIT will assign             **
4024  REM **    program disk units and select format for the           **
4028  REM **    output formats.                                        **
4032  REM **                                                           **
4036  REM **  ************************************************************
4040  DEFFN ' 200                                    : REM ENTRY
4044  GOSUB ' 201                                    : REM INITIALIZE TABLES
4048  GOSUB ' 202                                    : REM DISPLAY TITLE PAG
4052  GOSUB ' 203                                    : REM DISPLAY COMMANDS
4056  GOSUB ' 204                                    : REM EXECUTE COMMANDS
4060  RETURN                                         : REM EXIT
```

```
4064  REM **
4068  REM **
4072  REM **  Subroutine  ********************************************** 2 0 1
4076  REM **  Abstract >
4080  REM **    To initialize program Tables.
4084  REM **  Variables >
4088  REM **   R4$() > conversion tables
4092  REM **      (01)-(04) > <1,64>   lower case to upper case conv.
4096  REM **      (05)-(08) > <65,64>  upper case to lower case conv.
4100  REM **      (09)-(10) > <129,32> conv binary indx to fiche row char.
4104  REM **      (11) >      <161,16> control character conversion (input)
4108  REM **      (12) >      <177,16> buffer mode character conversions -
4112  REM **           >      <177,4>  HEX(00) to HEX(20)
4116  REM **           >      <181,4>  HEX(00) to HEX(00)
4120  REM **           >      <185,4>  HEX(30) to HEX(20)
4124  REM **           >      <189,4>  HEX(30) to HEX(00)
4128  REM **      (13)-(14) > <193,32> convert mask digits to normal codes
4132  REM **      (15)-(16) > <225,32> conv fiche row char. to binary indx
4136  REM **      (17)      > <257,16> input char conversion (entry mode)
4140  REM **      (18)-(20) > <273,48> input char conversion (edit mode)
4144  REM **   R5$() > single character search ids
4148  REM **      (01) > blank          (02) > HEX(FF)
4152  REM **      (03) > HEX(80)        (04) > HEX(5F)<underscor>
4156  REM **      (05) > HEX(3A)<coln>  (06) > HEX(3C) < < >
4160  REM **      (07) > HEX(3E)< > >   (08) > HEX(2E) < . >
4164  REM **      (09) > HEX(20)<blnk>  (10) > HEX(23) < # >
4168  REM **   R6$() > cusor control <vert.>
4172  REM **   R7$() > cusor control <horz.>
4176  REM **   K5$() > cusor control <erase>
4180  REM **   R8$() > micro command buffer & output sequences
4184  REM **      (01) > cusor blink w/  overprint char.
4188  REM **      (02) > string output with CRLF
4192  REM **      (03) >
4196  REM **      (04) > warning tone w/ 2 sec wait
4200  REM **      (05) >
4204  REM **      (06) > output 16 line buffer to crt
4208  REM **      (07) > single tone no wait
4212  REM **      (08) > setup register for CR or LF
4216  REM **      (09) > command register buffer
4220  REM **   Q0() > special register stack
4224  REM **      (01) > number of symbol sets
4228  REM **      (02) > number of symbls
4232  REM **      (03) > symbol table size <rows>
4236  REM **      (04) > symbol tabel size <cols>
4240  REM **      (05) > fiche pages <max>
4244  REM **      (06) > fiche rows <max>
4248  REM **      (07) > fiche cols <max>
4252  REM **      (08) > number of reserve index pages
4256  REM **      (09) > beging of output buffer
4260  REM **      (10) > size of output buffer
4264  REM **      (11) > number of criterion tests to generate
4268  REM **      (12) > flag - null crt display (0-clr 1-set)
```

```
4272  REM **    Q0$() > special program flags
4276  REM **       (01) > buffer contains incomplete item
4280  REM **       (02) > update output indx on mutiple volumes
4284  REM **       (03) > attemp to find smaller symbol
4288  REM **       (04) > current column of test page
4292  REM **       (05) > 1st page of sequence
4296  REM **       (06) > fiche page index (starting reference)
4300  REM **       (07) > fiche row  index (starting reference)
4304  REM **       (08) > fiche col  index (starting reference)
4308  REM **       (09) > auto right justify option (Y or N)
4312  REM **       (10) > auto left justify option (1 - 9)
4316  REM **    Q4$() > page address memory buffer
4320  REM **    Q5$() > keyword storage buffer
4324  REM **    Q6$() > format command index buffer
4328  REM **
4332  REM ************************************************
4336  DEFFN ' 201                                          : REM ENTRY
4340  R4$(01)=  HEX(41614262436344644565466647674868)
4344  R4$(02)=  HEX(49694A6A4B6B4C6C4D6D4E6E4F6F5070)
4348  R4$(03)=  HEX(5171527253735474557556765777587 8)
4352  R4$(04)=  HEX(59795A7A20202020202020202020202020)
4356  R4$(05)=  HEX(614162426343644465456646674768 48)
4360  R4$(06)=  HEX(69496A4A6B4B6C4C6D4D6E4E6F4F7050)
4364  R4$(07)=  HEX(7151725273537454755576567757785 8)
4368  R4$(08)=  HEX(79597A5A20202020202020202020202020)
4372  R4$(09)=  HEX(204142434445464748494A4B4C4D4E4F)
4376  R4$(10)=  HEX(20202020202020202020202020202020)
4380  R4$(11)=  HEX(010802A7030D03E620202020202020 20)
4384  R4$(12)=  HEX(200020200000202020302020003020 20)
4388  R4$(13)=  HEX(30103111321233133414351536163717)
4392  R4$(14)=  HEX(38183919200020202020202020202020)
4396  R4$(15)=  HEX(000102030405060708090A0B0C0D0E0F)
4400  R4$(16)=  HEX(20202020202020202020202020202020)
4404  R4$(17)=  HEX(010802A703E504E6050D2020202020 20)
4408  R4$(18)=  HEX(01A001080BCB025E0CCA03280DC70429)
4412  R4$(19)=  HEX(0EC3052A05C2062D06C4072B07C6090D)
4416  R4$(20)=  HEX(0AE5082F08CC0220202020202020202020)
```

```
4420  R5$(01)=  HEX(20)          :       R5$(02)= HEX(FF)
4424  R5$(03)=  HEX(80)          :       R5$(04)= HEX(5F)
4428  R5$(05)=  HEX(3A)          :       R5$(06)= HEX(3C)
4432  R5$(07)=  HEX(3E)          :       R5$(08)= HEX(2E)
4436  R5$(09)=  HEX(20)          :       R5$(10)= HEX(23)
4440  INIT(0A)  R6$()            :       R6$(01)= HEX(01)
4444  INIT(09)  R7$()            :       R7$(01)= HEX(0D)
4448  INIT(00)  U1$()            :       U1      = 1
4452  INIT(20)  K5$()            :       K5      = 1
4456  INIT(FF)  Q4$()            :       Q4      = 0
4460  R8$(01)=  HEX(400642604008121142504008113614541608114G)
4464  R8$(02)=  HEX(4006A000402140OD400A2020)
4468  R8$(03)=  HEX(FFFF2020)
4472  R8$(04)=  HEX(40071221400712214007202O)
4476  R8$(05)=  HEX(FFFF2020)
4480  R8$(06)=  HEX(A000400D14271608400A2020)
4484  R8$(07)=  HEX(400740072020)
4488  R8$(08)=  HEX(050D14671708050A2020)
4492  R8$(09)=  HEX(05163F20556677000000)
4496  Q0 (01)=  00              :       Q0 (02)=  00
4500  Q0 (03)=  75              :       Q0 (04)=  06
4504  Q0 (05)=  00              :       Q0 (06)=  00
4508  Q0 (07)=  00              :       Q0 (08)=  00
4512  Q0 (09)=  01              :       Q0 (10)=  1024
4516  Q0 (11)=  01              :       Q0 (12)=  00
4520  Q0$(01)=  HEX(30)         :       Q0$(02)=  HEX(30)
4524  Q0$(03)=  HEX(30)         :       Q0$(04)=  HEX(31)
4528  Q0$(05)=  HEX(30)         :       Q0$(06)=  HEX(01)
4532  Q0$(07)=  HEX(01)         :       Q0$(08)=  HEX(01)
4536  Q0$(09)=  HEX(4E)         :       Q0$(10)=  HEX(00)
4540  Q5$(01)=  "skb0"          :       Q5$(02)=  "skf0"
4544  Q5$(03)=  "indx"          :       Q5$(04)=  "clos"
4548  Q5$(05)=  "prac"          :       Q5$(06)=  "end*"
4552  Q5$(07)=  "dia0"          :       Q5$(08)=  "@@@@"
4556  Q5$(09)=  "@@@@"          :       Q5$(10)=  "@@@@"
4560  Q6$(01)=  "text"          :       Q6$(02)=  "word"
4564  Q6$(03)=  "lower"         :       Q6$(04)=  "caps"
4568  Q6$(05)=  "table"         :       Q6$(06)=  "subtitle"
4572  Q6$(07)=  "page"          :       Q6$(08)=  "line"
4576  Q6$(09)=  "space"         :       Q6$(10)=  "null"
4580  D1      =  0              :       D2      =  128
4584  RETURN                                          : REM EXIT
```

```
4588  REM **
4592  REM **
4596  REM **  Subroutine  ********************************************** 202
4600  REM       Function   >
4604  REM         To Print Header
4608  REM       Variables  >
4612  REM         R0$() > scratch buffer    !
4616  REM ********************************************
4620  DEFFN ' 202                                                    : REM ENTRY
4624  SELECT        PRINT 005(80)                                    : REM SEL OUT DSPLY
4628  PRINT         HEX(03)                                          : REM CLR SCREEN
4632  PRINT         "Program   >  AUTHOR/3"                          : REM PRNT PROC NAME
4636  PRINT         "Version   >  2.0"                               : REM PRNT VERSION
4640  PRINT         "Language  >  BASIC"                             : REM PRNT SOURCE LANG
4644  PRINT         "Machine   >  WANG 2000/1"                       : REM PRNT HARDWARE
4648  PRINT         "Abstract  >"                                    : REM PRNT SUBTITLE
4652  PRINTUSING    4676                                             : REM PRNT ABSTRACT
4656  PRINTUSING    4680                                             : REM PRNT ABSTRACT
4660  PRINTUSING    4684                                             : REM PRNT ABSTRACT
4664  SELECT        PRINT 405                                        : REM SEL OUT DSPLY
4668  GOSUB ' 205  (16)                                              : REM WAIT FOR RESPONCE
4672  REM **                                                         : REM IMAGE FORAMTS
4676 %        AUTHOR/3 will construct from the Phase II Data Base a
4680 %        Learning Module.  The Module to be Constructed is for:
4684 %                 I D E N T I F I N G    S Y M B O L S
4688  RETURN                                                         : REM EXIT
```

```
4692  REM **
4696  REM **
4700  REM **  Subroutine  ********************************************** 2 0 3
4704  REM **  Abstract >
4708  REM **    To display program options and commands.
4712  REM **  Variables >
4716  REM **    R2$ > input command    ! R0 > command indx
4720  REM **
4724  REM ************************************************
4728  DEFFN ' 203                                              : REM ENTRY
4732  DIM           W1$(4)2,W7$(4)1                            : REM CMMND TABLE
4736  IF            W7$(4) = HEX(2A)          THEN 4760        : REM TABL INITIZED
4740  INIT(20)      W7$()                                      : REM INIT TABLE
4744                W1$(1) = "AS"                              : REM INIT TABLE
4748                W1$(2) = "FO"                              : REM INIT TABLE
4752                W1$(3) = "SE"                              : REM INIT TABLE
4756                W1$(4) = "HE"                              : REM INIT TABLE
4760  SELECT        PRINT 005(80)                              : REM SEL OUT DISPLY
4764  PRINT         HEX(03)                                    : REM CLR SCREEN
4768  PRINTUSING    4876                                       : REM PRINT MENU
4772  PRINTUSING    4880, W7$(1)                               : REM PRINT MENU
4776  PRINTUSING    4884, W7$(2)                               : REM PRINT MENU
4780  PRINTUSING    4888                                       : REM PRINT MENU
4784  PRINTUSING    4892, W7$(3)                               : REM PRINT MENU
4788  PRINTUSING    4896                                       : REM PRINT MENU
4792  PRINTUSING    4900                                       : REM PRINT MENU
4796  SELECT        PRINT 405                                  : REM SEL OUT DSPLY
4800  GOSUB ' 101   (15,1,1,0)                                 : REM POSITION CUSUR
4804  PRINT         "Enter Command > "                         : REM PRNT PROMT
4808  GOSUB ' 206   (6,"A","HELP")                             : REM INPUT COMMAND
4812                R0 = LEN(R2$)                              : REM CMPUT CMND LEN
4816  IF            R0 <= 2                   THEN 4824        : REM WITHEN RANGE
4820                R0  = 2                                    : REM SET TO MAX LEN
4824  AND           (STR(R2$,1,R0),DF)                         : REM CONV LC TO UC
4828  MAT SEARCH    W1$(),=STR(R2$,1,R0) TO R2$() STEP 2:      REM SRCH CMND TBL
4832  IF            R2$(2) <> HEX(0000)       THEN 4840        : REM INVALID COMMAND
4836  IF            R2$(1) <> HEX(0000)       THEN 4860        : REM CMND FOUND
4840  GOSUB ' 101   (16,1,1,0)                                 : REM POSITION CUSUR
4844  PRINT         "INVALID COMMAND      RE-ENTER"            : REM PRNT ERR MSG
4848  $GIO          PAUSE TONE/005(R8$(4),R8$(9))              : REM PAUSE TONE
4852  GOSUB ' 101   (16,1,1,0)                                 : REM POSITION CUSUR
4856  GOTO          4800                                       : REM RE-ENTER CMND
4860                R0 = INT(FNR(VAL(R2$(1)))/2) + 1           : REM CMPUT CMND INDX
4864                W7$(R0) = HEX(2A)                          : REM STOR ITEM FLAG
4868                W7$(04) = HEX(2A)                          : REM SET INIT FLAG
4872  REM **                                                   : REM IMAGE FORAMTS
4876 % Command Summary;
4880 %  # ASSIGN - Assign Program and Data Disk units to the Program.
4884 %  # FORMAT - Select the formating to be used in Sequencing
4888 %             the the learning module.
4892 %  # SELECT - Select the Operation of Program (CREATE/LIST/EDIT)
4896 %    HELP   - Display text describing the Order in which
4900 %             to Execute the above Commands.
4904  RETURN                                                   : REM EXIT
```

```
4908  REM **
4912  REM **
4916  REM **  Subroutine  ********************************************** 2 0 4
4920  REM **  Abstract >
4924  REM **    To Process the inputted Command.
4928  REM **  Variables >
4932  REM **    R0  > command index  !
4936  REM **
4940  REM ***********************************************
4944  DEFFN ' 204                                          : REM ENTRY
4948  REM         <assign>  <format>  <select>  < help >: REM CMND FUNCTIONS
4952  ON R0 GOTO   4956,     4960,     4964,      4968  : REM BRANCH
4956  GOSUB ' 207                              : GOTO 4972 : REM ASSIGN DISKS
4960  GOSUB ' 208                              : GOTO 4972 : REM SEL FMT TYPE
4964  GOSUB ' 209                              : GOTO 4976 : REM SEL MODE
4968  GOSUB ' 210                              : GOTO 4972 : REM HELP
4972  GOSUB ' 203                              : GOTO 4952 : REM NEW CMND
4976  RETURN                                               : REM EXIT
```

```
4980  REM **
4984  REM **
4988  REM **  Subroutine  ********************************************** 2 0 5
4992  REM **  Abstract >
4996  REM **    To wait for operator responce displaying
5000  REM **    elapsed time of wait.
5004  REM **  Variables >
5008  REM **    S0$, S0 > time hours      ! K1 > row for display
5012  REM **    S1$, S1 > time mintues    !
5016  REM **    S2$, S2 > time seconds    !
5020  REM **    S3$, S3 > time adjustment!
5024  REM **
5028  REM ***********************************************
5032  DEFFN ' 205   (K1)                                    : REM ENTRY
5036  DIM           S0$2,S1$2,S2$2,S3$3                     : REM INITIALIZE
5040  GOSUB ' 101   (K1,1,1,0)                              : REM POSITION CUSOR
5044  PRINT         " Touch 'RETURN' to Continue"           : REM PRNT INST.
5048  FOR           S0 = 0 TO 2                             : REM BEG HOURS LOOP
5052  CONVERT       S0 TO S0$, (##)                         : REM CONV TO DISPLAY
5056  FOR           S1 = 0 TO 59                            : REM BEG MINS LOOP
5060  CONVERT       S1 TO S1$, (##)                         : REM CONV TO DISPLAY
5064  FOR           S2 = 0 TO 59                            : REM BEG SEC LOOP
5068  CONVERT       S2 TO S2$, (##)                         : REM CONV TO DISPLAY
5072  GOSUB ' 101   (K1,50,0,0)                             : REM POSITION CUSOR
5076  PRINT         S0$;":";S1$;":";S2$;HEX(06)             : REM PRNT CLOCK
5080  KEYIN         R0$(1), 5084, 5084      : GOTO 5088     : REM INPUT EOW
5084                S0=2 :  S1=59: S2=59    : GOTO 5096     : REM END OF LOOPS
5088  FOR           S3 = 0 TO 2150                          : REM BEG TIMMNG LOOP
5092  NEXT          S3                                      : REM CONT TIM CLOCK
5096  NEXT          S2                                      : REM CONT SEC CLOCK
5100  NEXT          S1                                      : REM CONT MIN CLOCK
5104  NEXT          S0                                      : REM CONT HRS CLOCK
5108  GOSUB ' 101   (K1,1,1,0)                              : REM POSITION CUSOR
5112  RETURN                                                : REM EXIT
```

```
5116  REM **
5120  REM **
5124  REM **  Subroutine  ********************************************* 2 0 6
5128  REM **  Abstract >
5132  REM **    Accept ALPHA and NUMERIC Input.
5136  REM **  Varaibles  >
5140  REM **    R0$() > entry buffer      ! R1  > max length
5144  REM **    R1$   > 'A' or 'N' or 'E'! R2  > return <num>
5148  REM **    R2$   > return <alph>     ! R3  > char pointer
5152  REM **    R6$   > initial value     ! R9  > scratch count
5156  REM **
5160  REM ************************************************
5164  DEFFN ' 206   (R1,R1$,R6$)                              : REM ENTRY
5168  INIT(3F)      R9$()                                     : REM INIT CHAR BUFR
5172  $GIO PRINT    /005 (A000,R8$(9))R9$()<1,R1>             : REM PRNT CUSOR BKGRND
5176  INIT(08)      R9$()                                     : REM INIT CHAR BUFR
5180  $GIO PRINT    /005 (A000,R8$(9))R9$()<1,R1>             : REM PRNT CUSOR BKGRND
5184  $GIO          /005 (1136,R8$(9))                        : REM INIT REGISTER
5188                R9  =  1                                  : REM PRNT POS INDX
5192                R3  =  1                                  : REM BUFR POS INDX
5196  INIT          (20) R2$                                  : REM CLR BUFR
5200 REM **
5204  KEYIN         R0$(1), 5204, 5204                        : REM CLR INPUT BUFR
5208                STR(R8$(9),5,1) = STR(R2$,R9,1)           : REM STOR OUT CHAR
5212  IF            R1$ <> "E"                  THEN 5220     : REM ECHO CHARACTR
5216                STR(R8$(9),5,1) = HEX(20)                 : REM SET NO ECHO
5220  KEYIN         R0$(1), 5260, 5236                        : REM INPUT CHAR
5224  $GIO          INPUT WAIT/005(R8$(1),R8$(9))             : REM PRNT CUSOR
5228  GOTO          5220                                      : REM CONT INPT WAIT
5232 REM **
5236  PRINT         HEX(0808);                                : REM FNT KEY ENTRY
5240  INPUT         R2$                                       : REM REENTER SPL KEY
5244                R3 = 5                                    : REM SET LENGTH
5248                STR(R2$,R3) = HEX(20)                     : REM CLR BUFR END
5252  GOTO          5480                                      : REM CONV IN DATA
5256 REM **
5260  IF            R0$(1) < HEX(08)            THEN 5280     : REM INVALID INPT
5264  $TRAN        (R0$()<1,1>, R4$()<161,10>) R              : REM CONV CTL CHARS
5268  IF           R0$(1) > HEX(7F)            THEN 5280     : REM INVALID INPT
5272  IF           R0$(1) > HEX(1F)            THEN 5300     : REM VALID CHAR
5276  IF           R0$(1) < HEX(04)            THEN 5300     : REM VALID CTL CHAR
5280  $GIO         TONE/005(R8$(7),R8$(9))                   : REM SOUND BELL
5284  GOTO         5220                                      : REM RE-ENTER CHAR
5288 REM **
5292 REM **                    ! RUBOUT ! UNDERSCORE ! ESCAPE !
5296 REM **                    !--------!------------!--------!
5300  ON VAL(R0$(1)) GOTO      5372  ,    5416     ,  5440   : REM BRANCH
5304 REM **                    !--------!------------!--------!
```

```
5308  REM **
5312  REM ********************************************** Enter Character
5316  REM **
5320  IF            R3 = R1 + 1                  THEN 5280  : REM BUFR FULL
5324                STR(R2$,R3,1) = R0$(1)                  : REM MOVE TO BUFR
5328                R3 = R3 + 1                             : REM INCR BUFR INDX
5332  IF            R3 = 2                       THEN 5344  : REM NOT FRST CHAR
5336                R9 = R9 + 1                             : REM INCR DSPLY INDX
5340  PRINT         HEX(09);                                : REM POSITION CUSOR
5344  IF            R1$ <> "E"                   THEN 5352  : REM ECHO CHARACTR
5348                R0$(1) = HEX(20)                        : REM SET NO ECHO
5352  PRINT         R0$(1);HEX(08);                         : REM PRNT CHAR
5356  GOTO          5208                                    : REM FECTH NXT CHAR
5360  REM **
5364  REM ********************************************** Rubout Character
5368  REM **
5372  IF            R3 = 1                       THEN 5208  : REM BUFR EMPTY
5376  PRINT         HEX(3F08);                              : REM POSITION CUSOR
5380                R3 = R3 - 1                             : REM DCRMNT BUF INDX
5384                STR(R2$,R3,1) = HEX(20)                 : REM CLR FRM BUFR
5388  IF            R9 = 1                       THEN 5400  : REM CUSOR POSITIONED
5392                R9 = R9 - 1                             : REM DCRMNT DSP INDX
5396  PRINT         HEX(08);                                : REM POSITION CUSOR
5400  GOTO          5208                                    : REM FETCH NXT CHAR
5404  REM **
5408  REM ********************************************** Underscore char.
5412  REM **
5416  AND           (STR(R2$,R9,1),7F)                      : REM MASK CHAR
5420  ADD           (STR(R2$,R9,1),80)                      : REM UNDERSCORE
5424  GOTO          5208                                    : REM FETCH NXT CHAR
5428  REM **
5432  REM ********************************************** String Entered
5436  REM **
5440  IF            R3 <> 1                      THEN 5468  : REM BUFR NOT EMPTY
5444                R2$ = R6$                               : REM INSRT DFLT VAL
5448  IF            R1$ <> "E"                   THEN 5456  : REM ECHO DFLT VAL
5452                R6$ = " "                               : REM CLR DFLT VAL
5456  PRINT         HEX(08);                                : REM POSITION CUSOR
5460  GOTO          5476                                    : REM BRNCH TO PRNT
5464  REM **
5468                STR(R2$,R3) = HEX(20)                   : REM CLR REST OF BUF
5472                R6$ = R2$                               : REM MOVE TO PRNT BUF
5476  PRINT         HEX(09);STR(R6$,R3,ABS(R1-R3)+1);       : REM PRNT STRNG
5480  IF            R1$ <> "N"                   THEN 5516  : REM RET ALPHA VAL
5484  IF            NUM(R2$) = 64                THEN 5512  : REM VALID NUM FMT
5488                R2  =  R1                               : REM SET CUSOR INDX
5492  IF            R3 <> R1+1                   THEN 5500  : REM INDX IS OK
5496                R2 =  R1+2                              : REM RESET INDX
5500  INIT(08) R9$(): $GIO/005(A000,R8$(9))R9$()<1,R2>      : REM PRNT CUSOR BKGRND
5504  $GIO          TONE/005(R8$(7),R8$(9))                 : REM ERR TONE
5508  GOTO          5168                                    : REM REENTER STRNG
5512  CONVERT       R2$ TO R2                               : REM SET NUMERIC VAL
5516  RETURN                                                : REM EXIT
```

```
5520  REM **
5524  REM **
5528  REM **  Subroutine  ********************************************** 2 0 7
5532  REM **  Abstract >
5536  REM **    To assign an input disk device to device table
5540  REM **    and verify disk has proper contents.
5544  REM **  Variables >
5548  REM **    R0 > eof flag          !
5552  REM **
5556  REM ***********************************************
5560  DEFFN ' 207                                          : REM ENTRY
5564  GOSUB ' 212                                          : REM INPUT DEV #
5568  GOSUB ' 213                                          : REM ASSIGN DEVICE
5572  GOSUB ' 214                                          : REM VERIFY CONTENTS
5576  IF          R0 <> 4                    THEN 5564     : REM NOT END OF CMND
5580  IF          R1 <> 6                    THEN 5564     : REM NOT END OF CMND
5584  RETURN                                               : REM EXIT
```

```
5588  REM **
5592  REM **
5596  REM **  Subroutine  ********************************************** 2 0 8
5600  REM **  Abstract >
5604  REM **    To display formatting menu and input command.
5608  REM **  Variables >
5612  REM **    R2$ > input command    ! R0 > command indx
5616  REM **
5620  REM **********************************************
5624  DEFFN ' 208                                          : REM ENTRY
5628  DIM           W4$(3)2                                : REM CMMND TABLE
5632                W4$(1) = "LI"                          : REM INIT TABLE
5636                W4$(2) = "24"                          : REM INIT TABLE
5640                W4$(3) = "48"                          : REM INIT TABLE
5644  SELECT        PRINT 005(80)                          : REM SEL OUT DISPLY
5648  PRINT         HEX(03)                                : REM CLR SCREEN
5652  PRINTUSING    5948                                   : REM PRINT MENU
5656  PRINTUSING    5952                                   : REM PRINT MENU
5660  PRINTUSING    5956                                   : REM PRINT MENU
5664  PRINTUSING    5960                                   : REM PRINT MENU
5668  SELECT        PRINT 405                              : REM SEL OUT DSPLY
5672  GOSUB ' 101   (15,1,1,0)                             : REM POSITION CUSOR
5676  PRINT         "Enter Format  > "                     : REM PRNT PROMT
5680  GOSUB ' 206   (6,"A","24x")                          : REM INPUT COMMAND
5684                R0  = LEN(R2$)                         : REM CMPUT CMND LEN
5688  IF            R0 <= 2                    THEN 5696   : REM WITHEN RANGE
5692                R0  = 2                                : REM SET TO MAX LEN
5696  AND           (STR(R2$,1,R0),DF)                     : REM FORCE UPPER CASE
5700  $TRAN         (STR(R2$,1,R0),R4$()<193,24>) R        : REM MASK DIGITS
5704  MAT SEARCH    W4$(),=STR(R2$,1,R0) TO R2$() STEP 2:  REM SRCH CMND TBL
5708  IF            R2$(2) <> HEX(0000)        THEN 5716   : REM INVALID COMMAND
5712  IF            R2$(1) <> HEX(0000)        THEN 5736   : REM CMND FOUND
5716  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
5720  PRINT         "INVALID COMMAND      RE-ENTER"        : REM PRNT ERR MSG
5724  $GIO          PAUSE TONE/005(R8$(4),R8$(9))          : REM PAUSE TONE
5728  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
5732  GOTO          5672                                   : REM RE-ENTER CMND
5736                R0 = INT(FNR(VAL(R2$(1)))/2) + 1       : REM CMPUT CMND INDX
```

```
5740  REM **
5744  REM ************************************************  PROCESS COMMAND
5748  REM **
5752  REM           <linear>   <24x>      <48x>           : REM CMMND DISCRPTN
5756  ON RO GOTO     5760,      5780,      5800           : REM BRANCH
5760                DS = 3                               : REM SET INDX MODE
5764                GO(5) = 7                            : REM MAX FICHE PAG'S
5768                GO(6) = 7                            : REM MAX FICHE ROW'S
5772                GO(7) = 14                           : REM MAX FICHE COL'S
5776  GOTO          5816                                 : REM BRNCH TO NXT STEP
5780                DS = 1                               : REM SET INDX MODE
5784                GO(5) = 7                            : REM MAX FICHE PAG'S
5788                GO(6) = 7                            : REM MAX FICHE ROW'S
5792                GO(7) = 14                           : REM MAX FICHE COL'S
5796  GOTO          5816                                 : REM BRNCH TO NXT STEP
5800                DS = 2                               : REM SET INDX MODE
5804                GO(5) = 4                            : REM MAX FICHE PAG'S
5808                GO(6) = 15                           : REM MAX FICHE ROW'S
5812                GO(7) = 18                           : REM MAX FICHE COL'S
5816  GOSUB ' 101   (RO+1,3,0,0)                         : REM POSITION CUSOR
5820  PRINT         HEX(2A)                              : REM PRINT ITEM FLAG
5824 REM **
5828 REM **           ----------         ----------        SELECT BEGING PAG NUM
5832 REM **
5836  GOSUB ' 220                                        : REM ENTR STRT PAG #
5840 REM **
5844 REM **           ----------         ----------        SELECT TEXT OPTION 1
5848 REM **
5852  GOSUB ' 101   (10,1,1,0)                           : REM POSITION CUSOR
5856  PRINTUSING    5968, "> ";                          : REM PRINT PROMPT
5860  GOSUB ' 206   (3,"A","No")                         : REM INPUT A RESPONCE
5864  AND           (R2$,DF)                             : REM FORCE UPPER CASE
5868                GO$(9) = R2$                         : REM STOR RESPONCE
5872 REM **
5876 REM **           ----------         ----------        SELECT TEXT OPTION 2
5880 REM **
5884  GOSUB ' 101   (12,1,0,0)                           : REM POSITION CUSOR
5888  PRINTUSING    5976, " " ;                          : REM PRINT PROMPT
5892  GOSUB ' 101   (11,1,0,0)                           : REM POSITION CUSOR
5896  PRINTUSING    5972, "> ";                          : REM PRINT PROMPT
5900  GOSUB ' 206   (1,"N","O")                          : REM INPUT A RESPONCE
5904                BIN(GO$(10)) = R2                    : REM STOR RESPONCE
```

```
5908 REM **
5912 REM **          ----------          ----------       SELECT NUMBR OF TESTS
5916 REM **
5920  GOSUB ' 101   (14,1,0,0)                          : REM POSITION CUSOR
5924  PRINTUSING    5964, "> ";                         : REM PRNT PROMT
5928  GOSUB ' 206   (2,"N","01")                        : REM INPUT VALUE
5932                GO(11) = R2                         : REM STOR VALUE
5936 REM **
5940  GOSUB ' 205   (16)                                : REM RSPONCE WAIT
5944  REM **                                            : REM IMAGE FORMATS
5948 % Command Summary:
5952 %     LINEAR   -  Format Output in a linear foramt.
5956 %     24x      -  Format Output in a 24x fiche format.
5960 %     48x      -  Format Output in a 48x fiche foramt.
5964 % Select the Number of Criterion Tests to Generate   ##
5968 % Select Auto Right Justification Option (Yes or NO) ##
5972 % Select Auto Left  Indention     Option   (0 - 9)   ##
5976 %  (0-off; 1-indent 1 space; 2-indent 2 spaces;...)   #
5980  RETURN                                            : REM EXIT
```

```
5984  REM **
5988  REM **
5992  REM **  Subroutine  ********************************************** 2 0 9
5996  REM **  Abstract >
6000  REM **    To display Operational menu and input command.
6004  REM **  Variables >
6008  REM **    R2$ > input command    ! R0 > command indx
6012  REM **
6016  REM ***********************************************
6020  DEFFN ' 209                                                  : REM ENTRY
6024  DIM           W5$(3)2                                        : REM CMMND TABLE
6028                W5$(1) = "CR"                                  : REM INIT TABLE
6032                W5$(2) = "ED"                                  : REM INIT TABLE
6036                W5$(3) = "LI"                                  : REM INIT TABLE
6040  SELECT        PRINT 005(80)                                  : REM SEL OUT DISPLY
6044  PRINT         HEX(03)                                        : REM CLR SCREEN
6048  PRINTUSING    6168                                           : REM PRINT MENU
6052  PRINTUSING    6172                                           : REM PRINT MENU
6056  PRINTUSING    6176                                           : REM PRINT MENU
6060  PRINTUSING    6180                                           : REM PRINT MENU
6064  SELECT        PRINT 405                                      : REM SEL OUT DSPLY
6068  GOSUB ' 101   (15,1,1,0)                                     : REM POSITION CUSOR
6072  PRINT         "Enter Command > "                             : REM PRNT PROMT
6076  GOSUB ' 206   (6,"A","CREATE")                               : REM INPUT COMMAND
6080                R0 = LEN(R2$)                                  : REM CMPUT CMND LEN
6084  IF            R0 <= 2                          THEN 6092     : REM WITHEN LIMITS
6088                R0  = 2                                        : REM SET TO MAX LEN
6092  AND           (STR(R2$,1,R0),DF)                             : REM CONV LC TO UC
6096  MAT SEARCH    W5$(),=STR(R2$,1,R0) TO R2$() STEP 2:          REM SRCH CMND TBL
6100  IF            R2$(2) <> HEX(0000)          THEN 6108         : REM INVALID COMMAND
6104  IF            R2$(1) <> HEX(0000)          THEN 6128         : REM CMND FOUND
6108  GOSUB ' 101   (16,1,1,0)                                     : REM POSITION CUSOR
6112  PRINT         "INVALID COMMAND      RE-ENTER"                : REM PRNT ERR MSG
6116  $GIO          PAUSE TONE/005(R8$(4),R8$(9))                  : REM PAUSE TONE
6120  GOSUB ' 101   (16,1,1,0)                                     : REM POSITION CUSOR
6124  GOTO          6068                                           : REM RE-ENTER CMND
6128                R0 = INT(FNR(VAL(R2$(1)))/2) + 1               : REM CMPUT CMND INDX
6132  REM **
6136  REM ***********************************************  PROCESS COMMAND
6140  REM **
6144  REM           <create>   <edit>    <list>                    : REM CMMND DISCRPTN
6148  ON R0 GOTO     6152,      6156,     6160                     : REM BRANCH
6152  GOSUB ' 102   ("Symbol Table","ATH3STAB","4000")             : REM LOAD PROC
6156  GOSUB ' 102   ("Edit"        ,"ATH3LIST","1500")             : REM LOAD PROC
6160  GOSUB ' 102   ("List"        ,"ATH3LIST","1500")             : REM LOAD PROC
6164  REM **                                                       : REM IMAGE FORMATS
6168 % Command Summary:
6172 %    CREATE   -  Begin Creation of the Learning Module.
6176 %    EDIT     -  Begin Editing of an Existing Learning Module.
6180 %    LIST     -  Begin Listing of an Existing Learning Module.
6184  RETURN                                                       : REM EXIT
```

```
4676  - 4652
4680  - 4656
4684  - 4660
4760  - 4736
4800  - 4856
4824  - 4816
4840  - 4832
4860  - 4836
4876  - 4768
4880  - 4772
4884  - 4776
4888  - 4780
4892  - 4784
4896  - 4788
4900  - 4792
4952  - 4972
4956  - 4952
4960  - 4952
4964  - 4952
4968  - 4952
4972  - 4956 4960 4968
4976  - 4964
5084  - 5080
5088  - 5080
5096  - 5084
5168  - 5508
5204  - 5204
5208  - 5356 5372 5400 5424
5220  - 5212 5228 5284
5236  - 5220
5260  - 5220
5280  - 5260 5268 5320
5300  - 5272 5276
5344  - 5332
5352  - 5344
5372  - 5300
5400  - 5388
5416  - 5300
5440  - 5300
5456  - 5448
5468  - 5440
5476  - 5460
5480  - 5252
5500  - 5492
5512  - 5484
5516  - 5480
5564  - 5576 5580
5672  - 5732
5696  - 5688
5716  - 5708
5736  - 5712
5760  - 5756
5780  - 5756
5800  - 5756
5816  - 5776 5796
5948  - 5652
5952  - 5656
5956  - 5660
```

5960 - 5664
5964 - 5924
5968 - 5856
5972 - 5896
5976 - 5888
6068 - 6124
6092 - 6084
6108 - 6100
6128 - 6104
6152 - 6148
6156 - 6148
6160 - 6148
6168 - 6048
6172 - 6052
6176 - 6056
6180 - 6060

AUTHOR System: Phase 3 - ATH3INT1 - V MAP

```
D1     - 4580
D2     - 4580
D5     - 5760 5780 5800
K1     - 5032 5040 5072 5108
K5     - 4452
K5$(   - 4452
Q0(    - 4496 4500 4504 4508 4512 4516 5764 5768 5772 5784 5788 5792 5804 5808
         5812 5932
Q0$(   - 4520 4524 4528 4532 4536 5868 5904
Q4     - 4456
Q4$(   - 4456
Q5$(   - 4540 4544 4548 4552 4556
Q6$(   - 4560 4564 4568 4572 4576
R0     - 4812 4816 4820 4824 4828 4860 4864 4952 5576 5684 5688 5692 5696 5700
         5704 5736 5756 5816 6080 6084 6088 6092 6096 6128 6148
R0$(   - 5080 5204 5220 5260 5264 5268 5272 5276 5300 5324 5348 5352
R1     - 5164 5172 5180 5320 5476 5488 5492 5496 5580
R1$    - 5164 5212 5344 5448 5480
R2     - 5488 5496 5500 5512 5904 5932
R2$    - 4812 4824 4828 5196 5208 5240 5248 5324 5384 5416 5420 5444 5468 5472
         5484 5512 5684 5696 5700 5704 5864 5868 6080 6092 6096
R2$(   - 4828 4832 4836 4860 5704 5708 5712 5736 6096 6100 6104 6128
R3     - 5192 5244 5248 5320 5324 5328 5332 5372 5380 5384 5440 5468 5476 5492
R4$(   - 4340 4344 4348 4352 4356 4360 4364 4368 4372 4376 4380 4384 4388 4392
         4396 4400 4404 4408 4412 4416 5264 5700
R5$(   - 4420 4424 4428 4432 4436
R6$    - 5164 5444 5452 5472 5476
R6$(   - 4440
R7$(   - 4444
R8$(   - 4460 4464 4468 4472 4476 4480 4484 4488 4492 4848 5172 5180 5184 5208
         5216 5224 5280 5500 5504 5724 6116
R9     - 5188 5208 5336 5388 5392 5416 5420
R9$(   - 5168 5172 5176 5180 5500
S0     - 5048 5052 5084 5104
S0$    - 5036 5052 5076
S1     - 5056 5060 5084 5100
S1$    - 5036 5060 5076
S2     - 5064 5068 5084 5096
S2$    - 5036 5068 5076
S3     - 5088 5092
S3$    - 5036
U1     - 4448
U1$(   - 4448
W1$(   - 4732 4744 4748 4752 4756 4828
W4$(   - 5628 5632 5636 5640 5704
W5$(   - 6024 6028 6032 6036 6096
W7$(   - 4732 4736 4740 4772 4776 4784 4864 4868
```

AUTHOR System: Phase 3 -- ATH3INT1 - ' MAP

```
???? DEFFN' 101
       - 4800 4840 4852 5040 5072 5108 5672 5716 5728 5816 5852 5884 5892 5920
         6068 6108 6120
???? DEFFN' 102
       - 6152 6156 6160
4040 DEFFN' 200
4336 DEFFN' 201
       - 4044
4620 DEFFN' 202
       - 4048
4728 DEFFN' 203
       - 4052 4972
4944 DEFFN' 204
       - 4056
5032 DEFFN' 205
       - 4668 5940
5164 DEFFN' 206
       - 4808 5680 5860 5900 5928 6076
5560 DEFFN' 207
       - 4956
5624 DEFFN' 208
       - 4960
6020 DEFFN' 209
       - 4964
???? DEFFN' 210
       - 4968
???? DEFFN' 212
       - 5564
???? DEFFN' 213
       - 5568
???? DEFFN' 214
       - 5572
???? DEFFN' 220
       - 5836
```

5.3.2 ATH3INT2

This section of the 'ATH3INIT' program contains subroutines 21Ø to 22Ø.

The remainder of this section presents the 'ATH3INT2' program listing and reference maps.

```
6400  %## Program > AUTHOR/3  (ATH3INT2)  Version 2.0  2200 T/VP  780321
6404  REM **  Subroutine  ********************************************** 2 1 0
6408  REM **  Abstract >
6412  REM **    Provide the basic sequence for initializing
6416  REM **    the Data Entry Process.
6420  REM **  Variables >
6424  REM **    RO$() > input char bufr !
6428  REM **
6432  REM ***********************************************
6436  DEFFN ' 210                                          : REM ENTRY
6440  SELECT        PRINT 005(80)                          : REM SEL OUT DSPLY
6444  PRINT         HEX(03)                                : REM CLR CRT SCRN
6448  PRINTUSING    6508                                   : REM PRINT MENU
6452  PRINTUSING    6512                                   : REM PRINT MENU
6456  PRINTUSING    6516                                   : REM PRINT MENU
6460  PRINTUSING    6520                                   : REM PRINT MENU
6464  PRINTUSING    6524                                   : REM PRINT MENU
6468  PRINTUSING    6528                                   : REM PRINT MENU
6472  PRINTUSING    6532                                   : REM PRINT MENU
6476  PRINTUSING    6536                                   : REM PRINT MENU
6480  PRINTUSING    6540                                   : REM PRINT MENU
6484  PRINTUSING    6544                                   : REM PRINT MENU
6488  PRINTUSING    6548                                   : REM PRINT MENU
6492  PRINTUSING    6552                                   : REM PRINT MENU
6496  PRINTUSING    6556                                   : REM PRINT MENU
6500  GOSUB ' 205  (16)                                    : REM REPONCE WAIT
6504  REM                                                  : REM IMAGE FORMATS
6508  %    Command                When To Use
6512  %  ==============   ==========================================
6516  %    ASSIGN           This command MUST be the FIRST command.
6520  %                     Its purpose is to assign the systems Input and
6524  %                     Output Devices.
6528  %                     ----------------------------------------------
6532  %    FORMAT           This command is SECOND, ONLY IF the Mode of
6536  %                     Operation is to be CREATE.  Its purpose is to
6540  %                     establish the indexing format of the module.
6544  %                     ----------------------------------------------
6548  %    SELECT           This command MUST be the LAST command. Its
5552  %                     purpose is to begin execution of the system
6556  %                     Overlay Programs.
6560  RETURN                                               : REM EXIT
```

```
6564  REM **
6568  REM **
6572  REM **  Subroutine  ********************************************** 2 1 2
6576  REM **  Abstract >
6580  REM **    To Input the device structure to be assigned.
6584  REM **  Variables >
6588  REM **    R2  > input numeric val ! R1 > device tbl indx
6592  REM **  Device Specifications >
6596  REM **    #1  > AUTHOR Source Programs
6600  REM **    #2  > AUTHOR Phase II Data Base
6604  REM **    #3  > AUTHOR Phase III Data Base
6608  REM **    #4  > Flopy Disk Formatting Drive
6612  REM **
6616  REM ***********************************************
6620  DEFFN ' 212                                          : REM ENTRY
6624  DIM           W2$(5)2,W6$(5,2)1                      : REM CMMND TABLE
6628  IF            W6$(5,1) = HEX(00)        THEN 6640    : REM CHCK FLG
6632  INIT(20)      W6$()                                  : REM CLR ITEM FLAGS
6636  GOTO          6644                                   : REM DISPLAY MENU
6640                W6$(5,1) = HEX(20)                     : REM CLR FLG
6644                W2$(1) = "PR"                          : REM INIT TABLE
6648                W2$(2) = "P2"                          : REM INIT TABLE
6652                W2$(3) = "P3"                          : REM INIT TABLE
6656                W2$(4) = "FO"                          : REM INIT TABLE
6660                W2$(5) = "EX"                          : REM INIT TABLE
6664  SELECT        PRINT 005(80)                          : REM SEL OUT DISPLY
6668  PRINT         HEX(03)                                : REM CLR CRT SCREEN
6672  PRINTUSING    6920, " "                              : REM PRINT MENU
6676  PRINTUSING    6924, " "                              : REM PRINT MENU
6680  PRINTUSING    6928, " "                              : REM PRINT MENU
6684  PRINTUSING    6932, W6$(1,1),W6$(1,2)                : REM PRINT MENU
6688  PRINTUSING    6936, W6$(2,1),W6$(2,2)                : REM PRINT MENU
6692  PRINTUSING    6940, W6$(3,1),W6$(3,2)                : REM PRINT MENU
6696  PRINTUSING    6944, W6$(4,1),W6$(4,2)                : REM PRINT MENU
6700  PRINTUSING    6948, " "                              : REM PRINT MENU
6704  SELECT        PRINT 405                              : REM SEL OUT DSPLY
6708  IF            W6$(5,1) <> HEX(20)       THEN 6808    : REM CHCK FLAG
6712  GOSUB ' 101   (15,1,1,0)                             : REM POSITION CUSOR
6716  PRINT         "Enter Command > "                     : REM PRNT PROMT
6720  GOSUB ' 206   (8,"A","EXIT")                         : REM INPUT COMMAND
6724                R0 = LEN(R2$)                          : REM CMPUT CMND LEN
6728  IF            R0 <= 2                    THEN 6736   : REM WITHEN LIMITS
6732                R0  = 2                                : REM SET TO MAX LEN
6736  AND           (STR(R2$,1,R0),DF)                     : REM FORCE UPPER CASE
6740  $TRAN         (STR(R2$,1,R0),R4$()<193,24>) R        : REM RESET DIGIT CHARS
6744  MAT SEARCH    W2$(),=STR(R2$,1,R0) TO R2$() STEP 2:  REM SRCH CMND TBL
6748  IF            R2$(2) <> HEX(0000)         THEN 6756 : REM CMND FOUND
6752  IF            R2$(1) <> HEX(0000)         THEN 6776 : REM CMND FOUND
6756  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
6760  PRINT         "INVALID COMMAND       RE-ENTER"       : REM PRNT ERR MSG
6764  $GID          PAUSE TONE/005(R8$(4),R8$(9))          : REM PAUSE TONE
6768  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
6772  GOTO          6712                                   : REM RE-ENTER CMND
6776                R0 = INT(FNR(VAL(R2$(1)))/2) + 1       : REM CMPUT DEV TYPE
```

```
6780  REM **
6784  REM ********************************************** INPUT DEV NUMBER
6788  REM **
6792                  W6$(R0,1) = HEX(2A)                        : REM SET ITEM FLAG
6796  IF              W6$(5,1) <> HEX(20)        THEN 6808       : REM CHECK FLAG
6800                  W6$(5,1)  = HEX(00)                        : REM SET FLAG
6804  GOTO            6664                                       : REM REPRINT MENU
6808  IF              R0 <> 5                    THEN 6824       : REM BRNCH TO EXIT
6812                  R0  = 4                                    : REM SET TO EXIT IDX
6816                  R1 =  6                                    : REM SET TO NULL BRNCH
6820  GOTO            6952                                       : REM BRANCH TO EXIT
6824  GOSUB ' 101     (10,1,1,0)                                 : REM POSITION CUSOR
6828  SELECT          PRINT 005(64)                              : REM SEL OUTPUT DISPLY
6832  PRINT           "  1 - Flopy Disk Drive #1    <310>"
6836  PRINT           "  2 - Flopy Disk Drive #2    <B10>"
6840  PRINT           "  3 - Flopy Disk Drive #3    <350>"
6844  PRINT           "  4 - Fixed Disk Plater      <320>"
6848  PRINT           "  5 - Removable Disk Plater  <B20>"
6852  SELECT          PRINT 405                                  : REM SEL OUTPUT DISPLY
6856  GOSUB ' 101     (16,1,1,0)
6860  PRINT           "Enter Disk Device to Assign > "
6864  CONVERT         R0 TO R0$(1), (#)                          : REM CMPUT DEFLT DEV
6868  GOSUB ' 206     (1,"N",R0$(1))                             : REM INPUT DEV #
6872  IF              R2 > 5                     THEN 6880       : REM INVALID INPUT
6876  IF              R2 > 0                     THEN 6900       : REM VALID INPUT
6880  GOSUB ' 101     (16,1,1,0)                                 : REM POSITION CUSOR
6884  PRINT           "INVALID COMMAND     RE-ENTER"             : REM PRNT ERR MSG
6888  $GIO            PAUSE TONE/005(R8$(4),R8$(9))              : REM WARNING TONE
6892  GOSUB ' 101     (16,1,1,0)                                 : REM POSITION CUSOR
6896  GOTO            6856                                       : REM RE-ENTER CMND
6900                  R1 = R2                                    : REM STOR DEV #
6904                  R2 = R0                                    : REM STOR DEV TYPE
6908  CONVERT         R1 TO W6$(R0,2), (#)                       : REM STOR DEV ASSIND
6912  GOSUB ' 101     (10,1,7,0)                                 : REM ERASE DISK MENU
6916  REM                                                        : REM IMAGE FORMATS
6920 % Command Summary; #
6924 %#  Command     Device Assigned     Function
6928 %# ---------    ----------------    ----------------------------
6932 %#  PROGRAM       Disk; #          Assign AUTHOR Program Disk
6936 %#  P2            Disk; #          Assign INPUT Data Base Disk
6940 %#  P3            Disk; #          Assign PRIMARY OUTPUT Disk
6944 %#  FORMAT        Disk; #          Assign SECONDARY OUTPUT Disk
6948 %  # EXIT                          Return to Command Menu
6952  RETURN                                                     : REM EXIT
```

```
6956  REM **
6960  REM **
6964  REM **  Subroutine  ********************************************** 2 1 3
6968  REM **  Abstract >
6972  REM **    To assign device to internal table slot.
6976  REM **  Variables >
6980  REM **    R2 > device type     ! R1 > device number
6984  REM **
6988  REM ********************************************
6992  DEFFN ' 213                                          : REM ENTRY
6996  REM **
7000  REM **        ! 310  ! B10  ! 350  ! 320  ! B20  ! EXIT !
7004  REM **        !------!------!------!------!------!------!
7008  ON R1 GOTO      7020 , 7044 , 7068 , 7092 , 7116 , 7140 : REM BRANCH
7012  REM **        !------!------!------!------!------!------!
7016  REM **
7020  ON R2 GOTO    7024, 7028, 7032, 7036                 : REM BRNCH TO DEV
7024  SELECT        #1 310                    : GOTO 7140 : REM ASSIGN #1
7028  SELECT        #2 310                    : GOTO 7140 : REM ASSIGN #2
7032  SELECT        #3 310                    : GOTO 7140 : REM ASSIGN #3
7036  SELECT        #4 310                    : GOTO 7140 : REM ASSIGN #4
7040  REM **
7044  ON R2 GOTO    7048, 7052, 7056, 7060                 : REM BRNCH TO DEV
7048  SELECT        #1 B10                    : GOTO 7140 : REM ASSIGN #1
7052  SELECT        #2 B10                    : GOTO 7140 : REM ASSIGN #2
7056  SELECT        #3 B10                    : GOTO 7140 : REM ASSIGN #3
7060  SELECT        #4 310                    : GOTO 7140 : REM ASSIGN #4
7064  REM **
7068  ON R2 GOTO    7072, 7076, 7080, 7084                 : REM BRNCH TO DEV
7072  SELECT        #1 350                    : GOTO 7140 : REM ASSIGN #1
7076  SELECT        #2 350                    : GOTO 7140 : REM ASSIGN #2
7080  SELECT        #3 350                    : GOTO 7140 : REM ASSIGN #3
7084  SELECT        #4 310                    : GOTO 7140 : REM ASSIGN #4
7088  REM **
7092  ON R2 GOTO    7096, 7100, 7104, 7108                 : REM BRNCH TO DEV
7096  SELECT        #1 320                    : GOTO 7140 : REM ASSIGN #1
7100  SELECT        #2 320                    : GOTO 7140 : REM ASSIGN #2
7104  SELECT        #3 320                    : GOTO 7140 : REM ASSIGN #3
7108  SELECT        #4 310                    : GOTO 7140 : REM ASSIGN #4
7112  REM **
7116  ON R2 GOTO    7120, 7124, 7128, 7132                 : REM BRNCH TO DEV
7120  SELECT        #1 B20                    : GOTO 7140 : REM ASSIGN #1
7124  SELECT        #2 B20                    : GOTO 7140 : REM ASSIGN #2
7128  SELECT        #3 B20                    : GOTO 7140 : REM ASSIGN #3
7132  SELECT        #4 310                    : GOTO 7140 : REM ASSIGN #4
7136  REM **
7140  RETURN                                               : REM EXIT
```

```
7144  REM **
7148  REM **
7152  REM **  Subroutine  ********************************************** 2 1 4
7156  REM **  Abstract >
7160  REM **    To verify contents of mounted disks.
7164  REM **  Variables >
7168  REM **    RO > disk type        !
7172  REM **
7176  REM ***********************************************
7180  DEFFN ' 214                                          : REM ENTRY
7184  REM **
7188  REM           <program>  <phase2>  <phase3> <null>
7192  ON RO GOTO     7200,       7204,     7208,   7212 : REM BRANCH
7196  REM **
7200  GOSUB ' 215                                : GOTO 7212 : REM VRFY PROG DSK
7204  GOSUB ' 216                                : GOTO 7212 : REM VRFY P2 DSK
7208  GOSUB ' 217                                : GOTO 7212 : REM VRFY P3 DSK
7212  RETURN                                               : REM EXIT
```

```
7216  REM **
7220  REM **
7224  REM **  Subroutine  ********************************************** 2 1 5
7228  REM **  Abstract >
7232  REM **    To verify that a standard WANG disk is mounted.
7236  REM **  Variables >
7240  REM **    Q9$() > i/o buffer      ! R9 > scratch register
7244  REM **
7248  REM ********************************************
7252  DEFFN ' 215                                          : REM ENTRY
7256  MAT REDIM     Q9$(32)8                               : REM SET I/O BUFR
7260  DATA LOAD     BA T #1, (0,R9)Q9$()                   : REM LOAD SYS SCTR
7264  IF           Q9$(2)=HEX(0000000000000000)THEN 7288:  REM SYS SECTOR
7268  GOSUB ' 101   (15,1,1,0)                            : REM POSITION CUSOR
7272  PRINT         "PROGRAM DISK IMPROPERLY FORMATTED!":  REM PRNT ERR MSG
7276  $GIO          TONE/005(400?,R8$(9))                  : REM ERR TONE
7280  GOSUB ' 205   (16)                                   : REM RESPONCE WAIT
7284  GOSUB ' 101   (15,1,2,0)                            : REM POSITION CUSOR
7288  RETURN                                               : REM EXIT
```

```
7292  REM **
7296  REM **
7300  REM **  Subroutine  ********************************************  2 1 6
7304  REM **  Abstract >
7308  REM **    To verify that the proper Phase2 Data Base
7312  REM **    Disk is Mounted.
7316  REM **  Variables >
7320  REM **    Q9$() > transfer bufr    ! R2  > adrs register
7324  REM **    R8$() > micro cmnd bufr ! R2$ > input str value
7328  REM **    R9$() > scratch bufr    !
7332  REM **
7336  REM ********************************************
7340  DEFFN ' 216                                                : REM ENTRY
7344  MAT REDIM     Q9$(64)4                                     : REM SET I/O BUFR
7348  DATA LOAD     BA T #2, (0,R9)Q9$()                         : REM LOAD 0 SCTR
7352  IF            Q9$(3) = HEX(EEEEEEEE)     THEN 7392         : REM VALID FORMAT
7356  GOSUB ' 101   (15,1,1,0)                                   : REM POSITION CUSOR
7360  PRINT         "INVALID PHASE 2 DATA BASE FORMAT!"          : REM PRNT ERR MSG
7364  $GIO          TONE/005(4007,R8$(9))                        : REM ERR TONE
7368  GOSUB ' 205   (16)                                         : REM RESPONCE WAIT
7372  GOSUB ' 101   (15,1,2,0)                                   : REM POSITION CUSOR
7376  GOTO          7448                                         : REM BRANCH TO EXIT
7380  REM **
7384  REM ********************************************** CMPR PASSWORDS
7388  REM **
7392  MAT COPY      Q9$()<65,16> TO R9$()<1,64>                  : REM FETCH TITLE
7396  GOSUB ' 101   (14,1,2,0)                                   : REM POSITION CUSOR
7400  PRINT         " Volume ID > ";R9$(1)                       : REM PRNT TITLE
7404  GOSUB ' 101   (15,1,0,0)                                   : REM POSITION CUSOR
7408  PRINT         " Enter Password > "                         : REM ENTR PROMT
7412  GOSUB ' 206   (8,"E","database")                           : REM INPUT PASWRD
7416  MAT COPY      Q9$()<25,8> TO R9$()<1,64>                   : REM FETCH PASWRD
7420  IF            R2$ = R9$(1)                THEN 7448        : REM VALID PASWRD
7424  IF            R2$=HEX(3C2074616567293E) THEN 7448          : REM VALID PASWRD
7428  GOSUB ' 101   (16,1,1,0)                                   : REM POSITION CUSOR
7432  PRINT         " INVALID PASSWORD RE-ENTER"                 : REM PRNT ERR MSG
7436  $GIO          PAUSE TONE/005(R8$(4),R8$(9))                : REM PAUSE TONE
7440  GOSUB ' 101   (16,1,1,0)                                   : REM POSITION CUSOR
7444  GOTO          7404                                         : REM RE-ENTER
7448  RETURN                                                     : REM EXIT
```

```
7452  REM **
7456  REM **
7460  REM **  Subroutine  ********************************************  2 1 7
7464  REM **  Abstract >
7468  REM **    To initialize and verify the Phase III
7472  REM **    Data Base disk.
7476  REM **  Variables >
7480  REM **    U3$() > command table  ! R2$ > input command
7484  REM **
7488  REM ********************************************
7492  DEFFN ' 217                                      : REM ENTRY
7496  DIM           W3$(2)2                            : REM CMMND TBL
7500                W3$(1) = "IN"                      : REM INIT TBL
7504                W3$(2) = "EX"                      : REM INIT TBL
7508  SELECT        PRINT 005                          : REM SEL OUT DSPLY
7512  PRINT         HEX(03)                            : REM CLR CRT SCRN
7516  PRINT         "Command Summary:"                 : REM PRNT TITLE
7520  PRINT         "    INITIAL   -  ";               : REM PRNT COMMAND
7524  PRINT         "Initialize Phase 3 Data Disk."    : REM PRNT FUNCTION
7528  PRINT         "    EXIT      -  ";               : REM PRNT COMMAND
7532  PRINT         "Phase 3 Data Disk Formatted."     : REM PRNT FUNCTION
7536  SELECT        PRINT 405                          : REM SEL OUT DSPLY
7540  GOSUB ' 101   (15,1,1,0)                         : REM POSITION CUSUR
7544  PRINT         "Enter Command > "                 : REM PRNT PROMT
7548  GOSUB ' 206   (7,"A","EXIT")                     : REM INPUT COMMAND
7552  AND           (R2$,DF)                           : REM CONV LC TO UC
7556  MAT SEARCH    W3$(), =STR(R2$,1,2) TO R2$() STEP 2: REM SRCH CMND TBL
7560  IF            R2$(1) <> HEX(0000)        THEN 7584 : REM CMND FOUND
7564  GOSUB ' 101   (16,1,1,0)                         : REM POSITION CUSUR
7568  PRINT         "INVALID COMMAND      RE-ENTER"    : REM PRNT ERR MSG
7572  $GIO          PAUSE TONE/005(R8$(4),R8$(9))      : REM PAUSE TONE
7576  GOSUB ' 101   (16,1,1,0)                         : REM POSITION CUSUR
7580  GOTO          7536                               : REM RE-ENTER CMND
7584                R1 = INT(FNR(VAL(R2$(1)))/2) + 1   : REM CMPUT CMND INDX
7588  REM **
7592  REM ********************************************  PROCESS DISK
7596  REM **
7600  REM           <initialize disk>   <verify disk>  : REM CONTRO FUNCTION
7604  ON R1 GOTO        7608,               7612       : REM BRANCH
7608  GOSUB ' 218                                      : REM INITIALIZE DISK
7612  GOSUB ' 219                                      : REM VERIFY DISK
7616  RETURN                                           : REM EXIT
```

```
7620  REM **
7624  REM **
7628  REM **  Subroutine  ********************************************** 2 1 8
7632  REM **  Abstract >
7636  REM **    To Initialize the Phase III Data Base Disk.
7640  REM **  Variables >
7644  REM **    Q9$() > i/o buffer    ! R2$ > return input bufr
7648  REM **    R9$() > scratch bufr  ! R7$ > temp storage bufr
7652  REM **    R0$ > indx sctr adrs  ! R3$ > vol id mask
7656  REM **    R4$ > sctr adrs incr  ! R5$ > vol id incr
7660  REM **    R8$ > vol id numbr    ! R8 > sctr adrs indx
7664  REM **    R9 > subsctr indx     ! R7 > scrth adrs registr
7668  REM **
7672  REM ***********************************************
7676  DEFFN ' 218                                          : REM ENTRY
7680  MAT REDIM    Q9$(64)4                                : REM SET I/O BUFR
7684  INIT         (00) Q9$()                              : REM CLR BUFR
7688               Q9$(01) = HEX(000F03FF)                 : REM SYSTEM BYTES
7692               Q9$(02) = HEX(04000000)                 : REM SYSTEM BYTES
7696               Q9$(03) = HEX(FFFFFFFF)                 : REM ATH 3 INDCTR
7700               Q9$(04) = HEX(FFFFFFFF)                 : REM ATH 3 INDCTR
7704               Q9$(05) = HEX(10000010)                 : REM INDX BYTES
7708               Q9$(06) = HEX(03FF0000)                 : REM INDX BYTES
7712               Q9$(07) = HEX(44415420)                 : REM FILE NAME
7716               Q9$(08) = HEX(46494C45)                 : REM FILE NAME
7720               Q9$(13) = HEX(10000000)                 : REM VOLUME NUMBR
7724  GOSUB ' 101  (14,1,1,0)                              : REM POSITION CUSOR
7728  PRINT        "Enter Disk Volume ID > "               : REM PRNT PROMT
7732  GOSUB ' 206  (16,"A","AUTHOR Data Base")             : REM INPUT TEXT
7736               R9$(1)= R2$                             : REM STOR VALUE
7740  MAT COPY     R9$()<1,16> TO Q9$()<65,16>             : REM STOR IN BUFR
7744  GOSUB ' 101  (15,1,1,0)                              : REM POSITION CUSOR
7748  PRINT        "Enter Password > "                     : REM PRNT PROMT
7752  GOSUB ' 206  (8,"E","database")                      : REM INPUT TEXT
7756               R7$    = R2$                            : REM STOR VALUE
7760  GOSUB ' 101  (15,1,1,0)                              : REM POSITION CUSOR
7764  PRINT        "Verify Password > "                    : REM PRNT PROMT
7768  GOSUB ' 206  (8,"E","database")                      : REM INPUT VALUE
7772  IF           R7$ <> R2$                  THEN 7744   : REM INVALID ENTRY
7776               R9$(1) = R2$                            : REM STOR VALUE
7780  MAT COPY     R9$()<1,8> TO Q9$()<25,8>               : REM STOR IN BUFR
7784  DATA SAVE    BA T #3, (0,R7)Q9$()                    : REM SAVE BUFR
```

```
7788  REM **
7792  REM ********************************************** CREATE INDEX
7796  REM **
7800  MAT REDIM     Q9$(128)2                                  : REM SET I/O BUFR
7804                R0$ = HEX(0010)                            : REM INIT ADRS
7808                R3$ = HEX(0FFF)                            : REM INIT ADRS MASK
7812                R4$ = HEX(0004)                            : REM INIT ADRS INCR
7816                R5$ = HEX(1000)                            : REM INIT VOL # INCR
7820                R8$ = HEX(1000)                            : REM INIT VOL #
7824  INIT          (FF) Q9$()                                 : REM INIT BUFR
7828  GOSUB ' 101   (14,1,2,0)                                 : REM POSITION CUSOR
7832  PRINT         "Initializing Disk Index     ";            : REM PRNT HEADER
7836  PRINT         "Sector          Sub-Sector Index"         : REM PRNT HEADER
7840  FOR           R8 = 1 TO 15                               : REM BEG LOOP R8
7844  GOSUB ' 101   (15,30,0,0)                                : REM POSITION CUSOR
7848  PRINT         R8                                         : REM PRNT SCRT ADRS
7852  GOSUB ' 101   (15,51,0,0)                                : REM POSITION CUSOR
7856  FOR           R9 = 2 TO 128 STEP 2                       : REM BEG LOOP R9
7860  CONVERT       R9 TO R2$, (###)                           : REM CONV TO DSPLY
7864  PRINT         R2$;HEX(080808)                            : REM PRNT SUBSCTR INDX
7868  AND           (R0$,R3$)                                  : REM CLR VOL NUMBR
7872  IF            R0$ <= HEX(03FC)          THEN 7884        : REM NOT END OF SEQ
7876                R0$ = HEX(0010)                            : REM BEG OF SEQ
7880  ADD           (R8$,R5$)                                  : REM INCR VOL #
7884  OR            (R0$,R8$)                                  : REM INSRT VOL #
7888                Q9$(R9) = R0$                              : REM INSRT SCTR ADRS
7892  ADDC          (R0$,R4$)                                  : REM INCR SCTR ADRS
7896  NEXT          R9                                         : REM CONT BUFR FILL
7900  DATA SAVE     BA T #3, (R8,R7)Q9$()                      : REM SAVE BUFR
7904  NEXT          R8                                         : REM CONT INDX FILL
7908  RETURN                                                   : REM EXIT
```

```
7912  REM **
7916  REM **
7920  REM **  Subroutine  ********************************************  2 1 9
7924  REM **
7928  REM **  Abstract >
7932  REM **    To verify that the proper Phase3 Data Base
7936  REM **    Disk is Mounted.
7940  REM **  Variables >
7944  REM **    Q9$() > transfer bufr    ! R2  > adrs register
7948  REM **    R8$() > micro cmnd bufr  ! R2$ > input str value
7952  REM **    R9$() > scratch bufr     !
7956  REM **
7960  REM ***********************************************
7964  DEFFN ' 219                                          : REM ENTRY
7968  MAT REDIM     Q9$(64)4                               : REM SET I/O BUFR
7972  DATA LOAD     BA T #3, (0,R9)Q9$()                   : REM LOAD 0 SCTR
7976  IF            Q9$(3) = HEX(FFFFFFFF)     THEN 8016   : REM VALID FORMAT
7980  GOSUB ' 101   (15,1,1,0)                             : REM POSITION CUSOR
7984  PRINT         "INVALID PHASE 3 DATA BASE FORMAT!"    : REM PRNT ERR MSG
7988  $GIO          TONE/005(4007,R8$(9))                  : REM ERR TONE
7992  GOSUB ' 205   (16)                                   : REM RESPONCE WAIT
7996  GOSUB ' 101   (15,1,2,0)                             : REM POSITION CUSOR
8000  GOTO          8072                                   : REM BRANCH TO EXIT
8004  REM **
8008  REM ********************************************** CMPR PASSWORDS
8012  REM **
8016  MAT COPY      Q9$()<65,16> TO R9$()<1,64>            : REM FETCH TITLE
8020  GOSUB ' 101   (14,1,2,0)                             : REM POSITION CUSOR
8024  PRINT         " Volume ID > ";R9$(1)                 : REM PRNT TITLE
8028  GOSUB ' 101   (15,1,0,0)                             : REM POSITION CUSOR
8032  PRINT         " Enter Password > "                   : REM ENTR PROMT
8036  GOSUB ' 206   (8,"E","database")                     : REM INPUT PASWRD
8040  MAT COPY      Q9$()<25,8> TO R9$()<1,64>             : REM FETCH PASWRD
8044  IF            R2$ = R9$(1)                THEN 8072  : REM VALID PASWRD
8048  IF            R2$=HEX(3C2074616567203E)   THEN 8072  : REM VALID PASWRD
8052  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
8056  PRINT         " INVALID PASSWORD RE-ENTER"           : REM PRNT ERR MSG
8060  $GIO          PAUSE TONE/005(R8$(4),R8$(9))          : REM PAUSE TONE
8064  GOSUB ' 101   (16,1,1,0)                             : REM POSITION CUSOR
8068  GOTO          8028                                   : REM RE-ENTER
8072  RETURN                                               : REM EXIT
```

```
8076 REM **
8080 REM **
8084 REM **  Subroutine  ********************************************** 2 2 0
8088 REM **  Abstract >
8092 REM **    To assign the starting frame number for the learning
8096 REM **    package.  The number can be entered in either fiche
8100 REM **    format or linear format.
8104 REM **  Variables >
8108 REM **    GO$() > system register stack ! R2$ > input value buffer
8112 REM **
8116 REM **
8120 REM ***********************************************
8124  DEFFN ' 220                                          : REM ENTRY
8128  GOSUB ' 101 (15,1,1,0)                               : REM POSITION CUSOR
8132  PRINT       "Do You Wish to override starting ";:    REM PRNT PROMPT
8136  PRINT       "Frame Address (Yes or No) > ";          : REM PRNT PROMPT
8140  GOSUB ' 206 (3,"A","NO")                             : REM INPUT RESPONCE
8144  AND         (R2$,DF)                                 : REM MASK OUT L/C
8148  GOSUB ' 101 (15,1,1,0)                               : REM CLR PROMPT
8152  IF          STR(R2$,1,1) <> "Y"       THEN 8284      : REM NO OVERRIDE
8156 REM **
8160  GOSUB ' 101 (15,1,1,0)                               : REM POSITION CUSOR
8164  PRINT       "How is Frame Address to be Entered";    : REM PRNT PROMPT
8168  PRINT       " (Linear or Fiche) > ";                 : REM PRNT PROMPT
8172  GOSUB ' 206 (6,"A","FICHE")                          : REM INPUT RESPONCE
8176  AND         (R2$,DF)                                 : REM MASK OUT L/C
8180  GOSUB ' 101 (15,1,1,0)                               : REM CLR PROMPT
8184  IF          STR(R2$,1,1) = "L"        THEN 8248      : REM IF LINEAR FMT
8188 REM **
8192  GOSUB ' 101 (6,1,0,0)                                : REM POSITION CUSOR
8196  PRINT       "Enter Starting Fiche Pag Number > ":    REM PRNT PROMPT
8200  GOSUB ' 206 (2,"N","01")                             : REM INPUT RESPONCE
8204              BIN(GO$(6)) = R2                         : REM STOR RESULT
8208  GOSUB ' 101 (7,1,0,0)                                : REM POSITION CUSOR
8212  PRINT       "Enter Starting Fiche Row Number > ":    REM PRNT PROMPT
8216  GOSUB ' 206 (2,"N","01")                             : REM INPUT RESPONCE
8220              BIN(GO$(7)) = R2                         : REM STOR RESULT
8224  GOSUB ' 101 (8,1,0,0)                                : REM POSITION CUSOR
8228  PRINT       "Enter Starting Fiche Col Number > ":    REM PRNT PROMPT
8232  GOSUB ' 206 (2,"N","01")                             : REM INPUT RESPONCE
8236              BIN(GO$(8)) = R2                         : REM STOR RESULT
8240  GOTO        8284                                     : REM BRANCH TO EXIT
8244 REM **
8248  GOSUB ' 101 (6,1,0,0)                                : REM POSITION CUSOR
8252  PRINT       "Enter Starting Frame Address > "        : REM PRNT PROMPT
8256  GOSUB ' 206 (4,"N","0001")                           : REM INPUT RESPONCE
8260  BIN(GO$(6))= INT((R2-1)/(GO(7)*GO(6))) + 1            : REM CMPUT PAG #
8264             R2 = R2 - ((VAL(GO$(6))-1)*(GO(7)*GO(6))):REM RESET VALUE
8268  BIN(GO$(7))= INT((R2-1)/(GO(7))) + 1                 : REM CMPUT ROW #
8272             R2 = R2 - ((VAL(GO$(7))-1)*(GO(7)))       : REM RESET VALUE
8276  BIN(GO$(8))= R2                                      : REM CMPUT COL #
8280 REM **
8284  RETURN                                               : REM EXIT
```

```
6508  - 6448
6512  - 6452
6516  - 6456
6520  - 6460
6524  - 6464
6528  - 6468
6532  - 6472
6536  - 6476
6540  - 6480
6544  - 6484
6548  - 6488
6552  - 6492
6556  - 6496
6564  - 0008
6640  - 6628
6644  - 6636
6664  - 6804
6712  - 6772
6736  - 6728
6756  - 6748
6776  - 0008 6752
6780  - 0008
6808  - 6708 6796
6824  - 6808
6856  - 6896
6880  - 6872
6900  - 6876
6920  - 6672
6924  - 6676
6928  - 6680
6932  - 6684
6936  - 6688
6940  - 6692
6944  - 6696
6948  - 6700
6952  - 0008 6820
7020  - 7008
7024  - 7020
7028  - 7020
7032  - 7020
7036  - 7020
7044  - 7008
7048  - 7044
7052  - 7044
7056  - 7044
7060  - 7044
7068  - 7008
7072  - 7068
7076  - 7068
7080  - 7068
7084  - 7068
7092  - 7008
7096  - 7092
7100  - 7092
7104  - 7092
7108  - 7092
7116  - 7008
7120  - 7116
```

```
7124  - 7116
7128  - 7116
7132  - 7116
7140  - 7008 7024 7028 7032 7036 7048 7052 7056 7060 7072 7076 7080 7084 7096
        7100 7104 7108 7120 7124 7128 7132
7200  - 7192
7204  - 7192
7208  - 7192
7212  - 7192 7200 7204 7208
7288  - 7264
7292  - 0008
7392  - 7352
7404  - 7444
7448  - 0008 7376 7420 7424
7536  - 7580
7584  - 7560
7608  - 7604
7612  - 7604
7620  - 0008
7744  - 7772
7784  - 0008
7788  - 0008
7884  - 7872
7908  - 0008
7912  - 0008
8016  - 7976
8028  - 8068
8072  - 0008 8000 8044 8048
8248  - 8184
8284  - 8152 8240
```

```
Q0(    - 8260 8264 8268 8272
Q0$(   - 8204 8220 8236 8260 8264 8268 8272 8276
Q9$(   - 7256 7260 7264 7344 7348 7352 7392 7416 7680 7684 7688 7692 7696 7700
         7704 7708 7712 7716 7720 7740 7780 7784 7800 7824 7888 7900 7968 7972
         7976 8016 8040
R0     - 6724 6728 6732 6736 6740 6744 6776 6792 6808 6812 6864 6904 6908 7192
R0$    - 7804 7868 7872 7876 7884 7888 7892
R0$(   - 6864 6868
R1     - 6816 6900 6908 7008 7584 7604
R2     - 6872 6876 6900 6904 7020 7044 7068 7092 7116 8204 8220 8236 8260 8264
         8268 8272 8276
R2$    - 6724 6736 6740 6744 7420 7424 7552 7556 7736 7756 7772 7776 7860 7864
         8044 8048 8144 8152 8176 8184
R2$(   - 6744 6748 6752 6776 7556 7560 7584
R3$    - 7808 7868
R4$    - 7812 7892
R4$(   - 6740
R5$    - 7816 7880
R7     - 7784 7900
R7$    - 7756 7772
R8     - 7840 7848 7900 7904
R8$    - 7820 7880 7884
R8$(   - 6764 6888 7276 7364 7436 7572 7988 8060
R9     - 7260 7348 7856 7860 7888 7896 7972
R9$(   - 7392 7400 7416 7420 7736 7740 7776 7780 8016 8024 8040 8044
W2$(   - 6624 6644 6648 6652 6656 6660 6744
W3$(   - 7496 7500 7504 7556
W6$(   - 6624 6628 6632 6640 6684 6688 6692 6696 6708 6792 6796 6800 6908
```

```
???? DEFFN' 101
       - 6712 6756 6768 6824 6856 6880 6892 6912 7268 7284 7356 7372 7396 7404
         7428 7440 7540 7564 7576 7724 7744 7760 7828 7844 7852 7980 7996 8020
         8028 8052 8064 8128 8148 8160 8180 8192 8208 8224 8248
???? DEFFN' 205
       - 6500 7280 7368 7992
???? DEFFN' 206
       - 6720 6868 7412 7548 7732 7752 7768 8036 8140 8172 8200 8216 8232 8256
6436 DEFFN' 210
6620 DEFFN' 212
6992 DEFFN' 213
7180 DEFFN' 214
7252 DEFFN' 215
       - 7200
7340 DEFFN' 216
       - 7204
7492 DEFFN' 217
       - 7208
7676 DEFFN' 218
       - 7608
7964 DEFFN' 219
       - 7612
8124 DEFFN' 220
```

5.4 ATH3STAB

The purpose of the 'ATH3STAB' program is to clear from memory all program text which is not needed for the construction process and to load in a new COMMOM table.

The remainder of this section contains the 'ATH3STAB' program listing.

AUTHOR System: Phase 3 - ATH3STAB - CLRPRG

```
4000 %## Program > AUTHOR/3   (ATH3STAB)  Version > 2.0  2200 T/VP  780321
4010 REM ** Abstract ***************************************************
4020 REM **                                                            **
4030 REM **   To load the symbol table into common memory              **
4040 REM **                                                            **
4050 REM ****************************************************************
4060 REM **
4070 LOAD DC T   #1,"ATH3VTAB" 100,299
4080 END
```

5.5 ATH3VTAB

The purpose of the 'ATH3VTAB' program is to define a new COMMOM table containing the variables necessary to construct the learning module. After such definition the program will load the symbol table initializing program.

The remainder of this section contains the 'ATH3VTAB' program listing.

```
100 %## Program > AUTHOR/3   (ATH3VTAB)  Version > 2.0   2200 T/VP  780321
105  REM ** Abstract ***************************************************************
110  REM **                                                                        **
115  REM **   To load the symbol table into common memory                          **
120  REM **                                                                        **
125  REM ************************************************************************
130  REM **
135  REM ***********************************************   DEFINE PROG STORAGE
140  REM **
145  COM      R4$(20)16,R5$(10)1,R6$(16)1,R7$(64)1     : REM GENERAL REGISTERS
150  COM      R8$(10)24,R9$(1)64                       : REM GENERAL REGISTERS
155  COM      B0,B1,B2,B3,B4,B8,B9,B1$1,B1$(20)1       : REM BIT CONTROL
160  COM      D1,D2,D3,D4,D5,D4$4,D5$2                 : REM OUTPUT PAG INDX
165  COM      D6$2,D7$4,D8$2,D9$2                      : REM OUTPUT PAG INDX
170  COM      K1,K2,K3,K4,K5$(1)64                     : REM CUSOR CONTROL
175  COM      L1$25,L2$8,L3$4                          : REM PROGRAM LOADING
180  COM      W1,W2,W3,W4,W5,W6,W1$2                   : REM FICHE PAG INDX
185  COM      U1$(15)1,U0,U1                           : REM SYMBOL INDXING
190  COM      R0,R1,R2,R3,R4,R5,R6,R7,R8,R9            : REM GENERAL REGISTERS
195  COM      R0$2,R1$20,R2$64,R3$2,R4$2,R5$2          : REM GENERAL BUFFERS
200  COM      R6$64,R7$64,R8$2,R9$64,E1$4              : REM GENERAL BUFFERS
205  COM      R0$(1)1,R1$(32)63,R2$(2)2,R3$(1)64       : REM GENERAL BUFFERS
210  COM      S1,S2,S3,S4,S5,S4$(1)64,S8$(1)4,S9$(1)4 : REM BUFR STORAGE INDX
215  COM      Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9            : REM GENERAL REGISTERS
220  COM      Q0(12),Q0$(10)1,Q1$(128)2,Q2$(16,64)1    : REM GENERAL BUFFERS
225  COM      Q4$(50)2,Q5$(10)3,Q6$(10)2               : REM GENERAL BUFFERS
230  COM      Q7$(75,2)1                               : REM GENERAL BUFFERS
235  COM      Q3$(75,6)2                               : REM DEFFN SBL TBL
240  DIM      Q9$(64)4                                 : REM SCRATCH BUFFER
245  %##           END OF COMMON STORAGE AREA          ##
250  LOAD DC 1     #1, "ATH3SR1S" 4000
255  END                                               : REM END OF STEP
```

5.6 ATH3SRTS

The purpose of the 'ATH3SRTS' program is to initialize the symbol table. After the symbol table has been loaded and sorted the program which controls the ordering of the learning module will be loaded.

The remainder of this section presents the 'ATH3SRTS' program listing and reference maps.

AUTHOR System: Phase 3 - ATH3SRTS - CNTRL

```
4000  %## Program > AUTHOR/3  (ATH3SRTS)   Version > 2.0 2200 T/VP  780321
4004  REM **  Abstract  ***************************************************
4008  REM **                                                              **
4012  REM **    Overlay ATH3SRTS will sort into memory the                **
4016  REM **    Symbol Table.                                             **
4020  REM **                                                              **
4024  REM ******************************************************************
4028  DEFFN ' 200                                    : REM ENTRY
4032  GOSUB ' 201                                    : REM INITIALIZE
4036  GOSUB ' 202                                    : REM LOAD SBL TBL
4040  GOSUB ' 203                                    : REM SORT SBL TBL
4044  GOSUB ' 205                                    : REM CMPT INDX PACS
4048  GOSUB ' 002                                    : REM BEG CREATE PHASE
4052  RETURN                                         : REM EXIT
```

```
4056  REM **
4060  REM **
4064  REM **  Subroutine  ********************************************** 2 0 1
4068  REM **  Abstract >
4072  REM **    To define and initialize sort tables.
4076  REM **  Variables >
4080  REM **    M$()  > internal merge buffer
4084  REM **    C1$() > symbol set memory table
4088  REM **    C2$() > symbol number memory table
4092  REM **    S$()  > sector sort buffer
4096  REM **    L$()  > locator buffer
4100  REM **    V1$() > merge vector 1
4104  REM **    V2$() > merge vector 2
4108  REM **    W$()  > sort work buffer
4112  REM **    Q9$() > general i/o buffer
4116  REM **    Q3$() > symbol table
4120  REM **    L(L) > linear to subscript conversion
4124  REM **
4128  REM ************************************************
4132  DEFFN ' 201                                          : REM ENTRY
4136  DIM           M$(10,64)4,S$(64)4,W$(64)2             : REM DEFINE BUFRS
4140  DIM           L$(160)2,V1$(11)1,V2$(10)2             : REM DEFINE BUFRS
4144  DIM           C1$(10)1,C2$(10)1,T$(1,64)4            : REM DEFINE BUFRS
4148  DEFFN L(L)=   INT(VAL(STR(L$(L),2,1))/4)+1           : REM FUNCTION L(L)
4152  DEFFN M(M)=   256*M + VAL(STR(M$(R8,R9),2,1))        : REM FUNCTION M(M)
4156  INIT          (20)M$()                               : REM INIT MRG BUFR
4160  INIT          (00)C1$()                              : REM INIT SET CNTR
4164  INIT          (00)C2$()                              : REM INIT SBL CNTR
4168  INIT          (FF)S$()                               : REM INIT SRT BUFR
4172  INIT          (00)L$()                               : REM INIT LOCATOR
4176  INIT          (01)V1$()                              : REM INIT VECTOR 1
4180  INIT          (00)V2$()                              : REM INIT VECTOR 2
4184  INIT          (00)Q3$()                              : REM INIT SBL TBL
4188  GOSUB ' 101   (5,10,3,0)                             : REM POSITION CURSOR
4192  PRINT         "INITIALIZING SYMBOL TABLE ..."        : REM PRNT TITLE
4196  RETURN                                               : REM EXIT
```

```
4200  REM **
4204  REM **
4208  REM **  Subroutine  ********************************************** 2 0 2
4212  REM **  Abstract >
4216  REM **    To Load the Merge buffer from the data base
4220  REM **    index. Each row is sorted.
4224  REM **  Variables >
4228  REM **    Q9$() > i/o buffer     ! S$() > sort buffer
4232  REM **    M$()  > merge buffer   ! L$() > locator buffer
4236  REM **    R1 > indx sctr adrs    ! R2 > subsctr indx
4240  REM **    R9 > scrtch ards rgstr! R8$ > search id
4244  REM **
4248  REM ****************************************************
4252  DEFFN ' 202                                              : REM ENTRY
4256                R8$ = HEX(03F2)                            : REM SET SRCH ID
4260  GOSUB ' 101   (8,10,0,0)                                 : REM POSITION CUSOR
4264  PRINT         "Load Internal Merge Buffer "              : REM PRNT TITLE
4268  GOSUB ' 101   (8,40,0,0)                                 : REM POSITION CUSOR
4272  PRINT         "Sector     Sub-Sector"                    : REM PRNT COL HEADS
4276  FOR           R1 = 1 TO 10                               : REM BEG INDX LOOP
4280  GOSUB ' 101   (9,42,0,0)                                 : REM POSITION CUSOR
4284  PRINT         R1                                         : REM PRNT SCTR ADRS
4288  GOSUB ' 101   (9,55,0,0)                                 : REM POSITION CUSOR
4292  DATA LOAD     BA T #2, (R1,R9)T$()                       : REM LOAD INDX SCTR
4296  MAT SEARCH    T$(), >= R8$ TO L$() STEP 4                : REM SRCH I/O BUFR
4300  FOR           R2 = 1 TO 65                               : REM BEG SUBINDX LOOP
4304  IF            L$(R2) <> HEX(0000)        THEN 4324       : REM END OF BUFR
4308  IF            R2 <> 1                    THEN 4316       : REM NOT BEG OF ROW
4312                V1$(R1) = HEX(FF)                          : REM SET TO EMPTY ROW
4316                R2 = 65                                    : REM SET TO END
4320  GOTO          4332                                       : REM BRNCH TO NXT LOOP
4324                BIN(STR(L$(R2),1,1)) = 1                   : REM CONV TO SUBSCRPT
4328                BIN(STR(L$(R2),2,1)) = FNL(R2)             : REM CONV TO SUBSCRPT
4332  CONVERT       R2 TO R4$, (##)                            : REM CONV TO DSPLY
4336  PRINT         R4$;HEX(0808)                              : REM PRNT SBSCTR INDX
4340  NEXT          R2                                         : REM CONT SUBINDX LOOP
4344  MAT MOVE      T$(), L$(1)   TO S$(1)                     : REM LOAD SORT BUFR
4348  MAT SORT      S$()          TO W$(), L$()                : REM SORT BUFR
4352  MAT MOVE      S$() , L$(1)  TO M$(R1,1)                  : REM LOAD MERGE BUFR
4356  INIT          (FF)S$()                                   : REM INIT SORT BUFR
4360  INIT          (00)L$()                                   : REM INIT LOCATOR BUFR
4364  NEXT          R1                                         : REM CONT INDX LOOP
4368  RETURN                                                   : REM EXIT
```

```
4372  REM **
4376  REM **
4380  REM **  Subroutine  ********************************************** 2 0 3
4384  REM **  Abstract >
4388  REM **    To Sort the merge buffer creating the sorted
4392  REM **    symbol table.
4396  REM **  Variables >
4400  REM **    M$() > merge buffer  ! V1$() > work vectr 1
4404  REM **    V2$() > work vectr 2 ! L$()  > locator buffer
4408  REM **    R1 > symbol tbl indx ! R2 > symbol tbl indx
4412  REM **    R3 > bufr out indx   ! R5 > elements merged
4416  REM **    R4 > merge end flag  ! R6 > sbl id tbl indx
4420  REM **
4424  REM ***********************************************
4428  DEFFN ' 203                                          : REM ENTRY
4432              R1 = 1                                   : REM SBL TBL ROW INDX
4436              R2 = 1                                   : REM SBL TBL COL INDX
4440              R3 = 1                                   : REM BUFR OUT INDX
4444              R6 = 0                                   : REM SBL ID INDX
4448  GOSUB ' 101 (8,10,2,0)                               : REM POSITION CUSOR
4452  PRINT       "Load Symbol Table From Merge Buffer":   REM PRNT TITLE
4456  GOSUB ' 101 (9,20,0,0)                               : REM POSITION CUSOR
4460  PRINT       "Symbol >"                               : REM PRNT TITLE
4464  GOSUB ' 101 (10,30,0,0)                              : REM POSITION CUSOR
4468  PRINT       "Set  Number  Item"                      : REM PRNT COL HEADS
4472  GOSUB ' 101 (11,30,0,0)                              : REM POSITION CUSOR
4476  MAT MERGE   M$() TO V1$(), V2$(), L$()               : REM BEG MERGE
4480  IF          L$(1) = HEX(0000)          THEN 4580     : REM ELEMENTS MERGED
4484              R4 = VAL(V1$(11))                        : REM FETCH MRG FLAG
4488  IF          R4 = 0                     THEN 4496     : REM LOCATOR FULL
4492              V1$(R4) = HEX(FF)                        : REM MRG ROW EMPTY
4496              R5 = 160                                 : REM ELEMENTS TO MOV
4500  MAT MOVE    M$()(3,2), L$(1), R5 TO Q3$(R1,R2)       : REM LOAD SBL TBL
4504              R0$ = HEX(FFFF)                          : REM SRCH CHARACTER
4508  MAT SEARCH  Q3$()<R3,900-R3+1>,=R0$TO R2$()STEP 2: REM SRCH FOR END
4512  IF          R2$(1) <> HEX(0000)        THEN 4528     : REM UNEVEN LOAD
4516  IF          R3 > 900                   THEN 4580     : REM BUFFER FULL
4520              R0$ = HEX(0000)                          : REM SET SRCH STRING
4524  GOTO        4508                                     : REM BRANCH TO SRCH
4528              R5 = INT((FNR(VAL(R2$(1)))-1)/2)         : REM CMPUT END POS
4532              R3 = R3 + R5*2                           : REM INCR BUFOUT INDX
4536  GOSUB ' 206                                          : REM INCR SBL CNTRS
4540  IF          R3 > 899                   THEN 4580     : REM SORT COMPLETE
4544              R1 = INT((R3-1)/12) + 1                  : REM INCR ROW INDX
4548              R2 = INT((R3 - (R1-1)*12)/2) + 1         : REM INCR COL INDX
4552              Q0(1) = VAL(C1$(1))                      : REM STOR # OF SETS
4556              Q0(2) = VAL(C2$(1))                      : REM STOR # OF SBLS
4560  GOSUB ' 101 (13,10,0,0)                              : REM POSITION CUSOR
4564  PRINT       "Symbols Sets > ";Q0(1);"      ";        : REM PRNT # OF SETS
4568  PRINT       "     Symbols > ";Q0(2)                  : REM PRNT # OF SBLS
4572  GOSUB ' 101 (11,30,0,0)                              : REM POSITION CUSOR
4576  GOTO        4476                                     : REM CONT MERGE
4580  RETURN                                               : REM EXIT
```

```
4584  REM **
4588  REM **
4592  REM **  Subroutine  ********************************************** 205
4596  REM **  Abstract >
4600  REM **    To compute the number of index pages needed
4604  REM **    at the end of each fiche page.
4608  REM **  Variables >
4612  REM **    M2 > sbl tbl indx     ! M5 > temp row indx
4616  REM **    M6 > out pag indx     ! M7 > temp line indx
4620  REM **    Q7$() > sbl id tbl    ! Q3$() > sbl table
4624  REM **    S1$ > current sbl set! Q9$() > i/o bufr
4628  REM **    R9$() > scratch bufr ! R0$ > scrtch byte rgstr
4632  REM **    Q0() > spcal regstrs !
4636  REM **
4640  REM ***********************************************
4644  DEFFN ' 205                                          : REM ENTRY
4648  DIM           S1$1                                   : REM DEFINE BUFRS
4652  MAT REDIM     Q9$(4)64                               : REM SET I/O BUFR
4656                M2 = 1                                 : REM SET SBL ID INDX
4660                M5 = 2                                 : REM SET TMP LIN INDX
4664                M6 = 10                                : REM SET PAG ROW INDX
4668                M7 = 1                                 : REM SET BUFR INDX
4672  GOSUB ' 101   (5,10,7,0)                             : REM POSITION CUSOR
4676  PRINT         "COMPUTE NUMBER OF INDEX PAGES..."     : REM PRNT HEADING
4680  GOSUB ' 101   (7,15,0,0)                             : REM POSITION CUSOR
4684  PRINT         "Number of Index Pages > "             : REM PRNT TITLE
4688  PRINT         "01";HEX(0808)                         : REM PRNT INIT #
4692                S1$ = Q7$(M2,1)                        : REM SET BEG SET #
4696  IF            Q7$(M2,1) <> S1$          THEN 4736    : REM NEW SBL SET
4700  DATA LOAD     BA T #2, (Q3$(M2,3),R0$)Q9$()          : REM FETCH SYMBOL
4704  MAT COPY      Q9$() <1,15> TO R9$()<1,64>            : REM STOR SYMBOL
4708  IF            M7+LEN(R9$(1))+3 < 41     THEN 4720    : REM CONT SAME LINE
4712                M5 = M5 + 1                            : REM INCR LIN COUNT
4716                M7 = 1                                 : REM RESET LIN INDX
4720                M7 = M7 + LEN(R9$(1)) + 3              : REM INCR LIN INDX
4724                M2 = M2 + 1                            : REM INCR SBL INDX
4728  IF            M2 > Q0(2)                THEN 4736    : REM END OF SBL TBL
4732  GOTO          4696                                   : REM NEXT SYMBOL
4736  IF            M5 <= 16-M6               THEN 4756    : REM NOT E-O-P
4740                Q0(8) = Q0(8) + 1                      : REM INCR TOTL PAGS
4744  CONVERT       Q0(8)+1 TO R0$, (##)                   : REM CONV TO DSPLY
4748  PRINT         R0$;HEX(0808)                          : REM PRNT VALUE
4752                M6 = 3                                 : REM RESET LIN INDX
4756                M6 = M6 + M5                           : REM INCR LIN INDX
4760                M5 = 2                                 : REM RESET TMP INDX
4764                M7 = 1                                 : REM RESET LIN INDX
4768  IF            M2 > Q0(2)                THEN 4776    : REM SBL TBL EMPTY
4772  GOTO          4692                                   : REM NEXT SBL SET
4776  IF            16-M6 >= 7                THEN 4784    : REM INDX COMPLETE
4780                Q0(8) = Q0(8) + 1                      : REM INCR TOT PAGES
4784                Q0(8) = Q0(8) + 1                      : REM ADJUST PAG COUNT
4788  RETURN                                               : REM EXIT
```

```
4792  REM **
4796  REM **
4800  REM **  Subroutine  ********************************************** 206
4804  REM **  Abstract >
4808  REM **    To count the number of symbol sets and number
4812  REM **    of symbols.
4816  REM **  Variables >
4820  REM **    R5 > # of items merged ! R0 > sub-item number
4824  REM **    R8 > set number        ! R9 > symbl number
4828  REM **    R7 > packed ref #      ! R2$ > scratch bufr
4832  REM **    B1$ > temp bit patern  ! C1$() > cnt buf sets
4836  REM **    C2$() > cnt buf sbls   ! B1$() > intrnl mem tbl
4840  REM **    L$() > locator bufr    ! M$() > merge buffer
4844  REM **    R6   > sbl id tbl indx ! Q7$()> sbl id tbl
4848  REM **
4852  REM ***********************************************
4856  DEFFN ' 206                                          : REM ENTRY
4860  IF          R5 = 0                          THEN 4980 : REM MRC COMPLETED
4864  FOR         R4 = 1 TO R5                              : REM BEG COUNT LOOP
4868              R8 = VAL(STR(L$(R4),1,1))                 : REM CMPUT ROW SUBSCRPT
4872              R9 = VAL(STR(L$(R4),2,1))                 : REM CMPUT COL SUBSCRPT
4876              R7 = FNM(VAL(M$(R8,R9)))                  : REM FETCH REF #
4880              R8 = INT(R7/1000)                         : REM CMPUT SET #
4884              R9 = INT((R7-(R8*1000))/10)               : REM CMPUT SBL #
4888              R0 = R7 - (R8*1000 + R9*10)               : REM CMPUT ITM #
4892  INIT        (20)R2$                                   : REM INIT DSPLY BUFR
4896  CONVERT     R8 TO STR(R2$,1,2), (##)                  : REM CONV TO DSPLY
4900  CONVERT     R9 TO STR(R2$,8,2), (##)                  : REM CONV TO DSPLY
4904  CONVERT     R0 TO STR(R2$,15,2), (##)                 : REM CONV TO DSPLY
4908  PRINT       STR(R2$,1,16)                             : REM PRNT DSPLY BUFR
4912  INIT        (08)R2$                                   : REM INIT DSPLY BUFR
4916  PRINT       STR(R2$,1,16)                             : REM PRNT DSPLY BUFR
4920  MAT COPY    C1$() TO B1$()                            : REM STOR MEM TBL
4924  GOSUB  207  (B1$,B1$(1),R8)                           : REM CHECK MEM TBL
4928  IF          B1$ <> HEX(00)                  THEN 4944 : REM SET COUNTED
4932  MAT COPY    B1$() TO C1$()                            : REM STOR MEM TBL
4936              BIN(C1$(1)) = VAL(C1$(1)) + 1             : REM INCR COUNT
4940  $TRAN       (C2$()<2,9>, R4$()<181,4>)00 R            : REM ZERO SBL CNT
4944  MAT COPY    C2$() TO B1$()                            : REM STOR MEM TBL
4948  GOSUB ' 207 (B1$,B1$(1),R9)                           : REM CHECK MEM TBL
4952  IF          B1$ <> HEX(00)                  THEN 4976 : REM SET COUNTED
4956  MAT COPY    B1$() TO C2$()                            : REM STOR MEM TBL
4960              BIN(C2$(1)) = VAL(C2$(1)) + 1             : REM INCR COUNT
4964              R6 = R6 + 1                               : REM INCR ID INDX
4968              BIN(Q7$(R6,1)) = R8                       : REM STOR SET #
4972              BIN(Q7$(R6,2)) = R9                       : REM STOR SBL #
4976  NEXT        R4                                        : REM CONT COUNTING
4980  RETURN                                                : REM EXIT
```

```
4984  REM **
4988  REM **
4992  REM **  Subroutine  ********************************************** 2 0 7
4996  REM **  Abstract >
5000  REM **   To set a bit on corsponding to the item number.
5004  REM **   If bit is on return a zero bit patern.
5008  REM **  Variables >
5012  REM **    B1$ > temp storage byte ! B2 > item number
5016  REM **    B1$() > memory table    ! B0 > scratch indx
5020  REM **    B4 > bit position       !
5024  REM **
5028  REM ***********************************************
5032  DEFFN ' 207   (B1$,B1$(1),B2)                          : REM ENTRY
5036                B0 = INT((B2-1)/8) + 2                   : REM CMPUT ELEMNT
5040                B4 = B2 - (B0-2)*8                       : REM CMPUT BIT POS
5044  GOSUB ' 208   (B1$(B0),B4)                             : REM MASK BYTE
5048  IF            B1$ <> HEX(00)            THEN 5064      : REM BIT WAS ON
5052  GOSUB ' 209   (B1$(B0),B4)                             : REM TURN BIT ON
5056                B1$(B0) = B1$                            : REM STOR NEW BIT PTRN
5060                B1$ = HEX(00)                            : REM RESET FLAG
5064  RETURN                                                 : REM EXIT
```

```
5068  REM **
5072  REM **
5076  REM **  Subroutine  ********************************************** 2 0 8
5080  REM **  Abstract >
5084  REM **    To mask a given bit position.
5088  REM **  Variables >
5092  REM **    B1$ > storage byte    ! B4 > bit position
5096  REM **
5100  REM ***********************************************
5104  DEFFN ' 208  (B1$,B4)                                  : REM ENTRY
5108  REM **
5112  REM **       ! BIT 80 ! BIT 40 ! BIT 20 ! BIT 10 !
5116  REM **       !--------!--------!--------!--------!
5120  ON B4 GOTO     5136  ,  5140  ,  5144  ,  5148  ,
                     5152  ,  5156  ,  5160  ,  5164  : REM BRANCH
5124  REM **       !--------!--------!--------!--------!
5128  REM **       ! BIT 04 ! BIT 04 ! BIT 02 ! BIT 01 !
5132  REM **
5136  AND          (B1$,80)                 : GOTO 5168 : REM MASK BIT 8
5140  AND          (B1$,40)                 : GOTO 5168 : REM MASK BIT 7
5144  AND          (B1$,20)                 : GOTO 5168 : REM MASK BIT 6
5148  AND          (B1$,10)                 : GOTO 5168 : REM MASK BIT 5
5152  AND          (B1$,08)                 : GOTO 5168 : REM MASK BIT 4
5156  AND          (B1$,04)                 : GOTO 5168 : REM MASK BIT 3
5160  AND          (B1$,02)                 : GOTO 5168 : REM MASK BIT 2
5164  AND          (B1$,01)                 : GOTO 5168 : REM MASK BIT 1
5168  RETURN                                             : REM EXIT
```

```
5172  REM **
5176  REM **
5180  REM **  Subroutine  ********************************************** 2 0 9
5184  REM **  Abstract >
5188  REM **    To turn on a given bit position.
5192  REM **  Variables >
5196  REM **    B1$ > storage byte    ! B4 > bit position
5200  REM **
5204  REM ********************************************
5208  DEFFN ' 209  (B1$,B4)                                  : REM ENTRY
5212  REM **
5216  REM **       ! BIT 80 ! BIT 40 ! BIT 20 ! BIT 10 !
5220  REM **       !--------!--------!--------!--------!
5224  ON B4 GUTU     5240  ,  5244  ,  5248  ,  5252  ,
                     5256  ,  5260  ,  5264  ,  5268  : REM BRANCH
5228  REM **       !--------!--------!--------!--------!
5232  REM **       ! BIT 04 ! BIT 04 ! BIT 02 ! BIT 01 !
5236  REM **
5240  OR           (B1$,80)                 : GOTO 5272 : REM MASK BIT 8
5244  OR           (B1$,40)                 : GOTO 5272 : REM MASK BIT 7
5248  OR           (B1$,20)                 : GOTO 5272 : REM MASK BIT 6
5252  OR           (B1$,10)                 : GOTO 5272 : REM MASK BIT 5
5256  OR           (B1$,08)                 : GOTO 5272 : REM MASK BIT 4
5260  OR           (B1$,04)                 : GOTO 5272 : REM MASK BIT 3
5264  OR           (B1$,02)                 : GOTO 5272 : REM MASK BIT 2
5268  OR           (B1$,01)                 : GOTO 5272 : REM MASK BIT 1
5272  RETURN                                             : REM EXIT
```

4316 - 4308
4324 - 4304
4332 - 4320
4476 - 4576
4496 - 4488
4508 - 4524
4528 - 4512
4580 - 4480 4516 4540
4692 - 4772
4696 - 4732
4720 - 4708
4736 - 4696 4728
4756 - 4736
4776 - 4768
4784 - 4776
4944 - 4928
4976 - 4952
4980 - 4860
5064 - 5048
5136 - 5120
5140 - 5120
5144 - 5120
5148 - 5120
5152 - 5120
5156 - 5120
5160 - 5120
5164 - 5120
5168 - 5136 5140 5144 5148 5152 5156 5160 5164
5240 - 5224
5244 - 5224
5248 - 5224
5252 - 5224
5256 - 5224
5260 - 5224
5264 - 5224
5268 - 5224
5272 - 5240 5244 5248 5252 5256 5260 5264 5268

```
 B0     - 5036 5040 5044 5052 5056
 B1$    - 4924 4928 4948 4952 5032 5048 5056 5060 5104 5136 5140 5144 5148 5152
          5156 5160 5164 5208 5240 5244 5248 5252 5256 5260 5264 5268
 B1$(   - 4920 4924 4932 4944 4948 4956 5032 5044 5052 5056
 B2     - 5032 5036 5040
 B4     - 5040 5044 5052 5104 5120 5208 5224
 C1$(   - 4144 4160 4552 4920 4932 4936
 C2$(   - 4144 4164 4556 4940 4944 4956 4960
 L      - 4148
 L$(    - 4140 4148 4172 4296 4304 4324 4328 4344 4348 4352 4360 4476 4480 4500
          4868 4872
 M      - 4152
 M$(    - 4136 4152 4156 4352 4476 4500 4876
 M2     - 4656 4692 4696 4700 4724 4728 4768
 M5     - 4660 4712 4736 4756 4760
 M6     - 4664 4736 4752 4756 4776
 M7     - 4668 4708 4716 4720 4764
 Q0(    - 4552 4556 4564 4568 4728 4740 4744 4768 4780 4784
 Q3$(   - 4184 4500 4508 4700
 Q7$(   - 4692 4696 4968 4972
 Q9$(   - 4652 4700 4704
 R0     - 4888 4904
 R0$    - 4504 4508 4520 4700 4744 4748
 R1     - 4276 4284 4292 4312 4352 4364 4432 4500 4544 4548
 R2     - 4300 4304 4308 4316 4324 4328 4332 4340 4436 4500 4548
 R2$    - 4892 4896 4900 4904 4908 4912 4916
 R2$(   - 4508 4512 4528
 R3     - 4440 4508 4516 4532 4540 4544 4548
 R4     - 4484 4488 4492 4864 4868 4872 4976
 R4$    - 4332 4336
 R4$(   - 4940
 R5     - 4496 4500 4528  532 4860 4864
 R6     - 4444 4964 4968 4972
 R7     - 4876 4880 4884 4888
 R8     - 4152 4868 4876 4880 4884 4888 4896 4924 4968
 R8$    - 4256 4296
 R9     - 4152 4292 4872 4876 4884 4888 4900 4948 4972
 R9$(   - 4704 4708 4720
 S$(    - 4136 4168 4344 4348 4352 4356
 S1$    - 4648 4692 4696
 T$(    - 4144 4292 4296 4344
 V1$(   - 4140 4176 4312 4476 4484 4492
 V2$(   - 4140 4180 4476
 W$(    - 4136 4348
```

```
???? DEFFN' 2
       - 4048
???? DEFFN' 101
       - 4188 4260 4268 4280 4288 4448 4456 4464 4472 4560 4572 4672 4680
4028 DEFFN' 200
4132 DEFFN' 201
       - 4032
4252 DEFFN' 202
       - 4036
4428 DEFFN' 203
       - 4040
4644 DEFFN' 205
       - 4044
4856 DEFFN' 206
       - 4536
5032 DEFFN' 207
       - 4924 4948
5104 DEFFN' 208
       - 5044
5208 DEFFN' 209
       - 5052
```

5.7 ATH3AUTO

The purpose of the 'ATH3AUTO' program is to provide the overlay control logic for the major construction of the learning module. This program also contains utility routines for generating a random subscript and for loading a format from disk into the page buffer.

The remainder of this section presents the 'ATH3AUTO' program listing and reference maps.

```
4000  %## Program > AUTHOR/111  (ATH3AUTO)  Version > 2.0 2200 1/VP  780321
4004  REM **  Abstract  ************************************************
4008  REM **                                                            **
4012  REM **    To control the construction of the Symbol               **
4016  REM **    Learning Package.                                       **
4020  REM **                                                            **
4024  REM ************************************************************************
4028  DEFFN ' 017                                                  : REM ENTRY 017
4032  GOSUB ' 102 ("Title Creation","ATH300.0","6000")             : REM LOAD
4036  DEFFN ' 018                                                  : REM ENTRY 018
4040  GOSUB ' 102 ("Introduction Creation","ATH301.0","6000")      : REM LOAD
4044  DEFFN ' 019                                                  : REM ENTRY 019
4048  GOSUB ' 102 ("Importance Creation","ATH302.0","6000")        : REM LOAD
4052  DEFFN ' 020                                                  : REM ENTRY 020
4056  GOSUB ' 102 ("Example Creation","ATH303.0","6000")           : REM LOAD
4060  DEFFN ' 021                                                  : REM ENTRY 021
4064  GOSUB ' 102 ("Pre-Test Creation","ATH304.0","6000")          : REM LOAD
4068  DEFFN ' 022                                                  : REM ENTRY 022
4072  GOSUB ' 102 ("Optional Test Creation","ATH304.1","6000")     : REM LOAD
4076  DEFFN ' 023                                                  : REM ENTRY 023
4080  GOSUB ' 102 ("Lesson chart Creation","ATH305.0","6000")      : REM LOAD
4084  DEFFN ' 024                                                  : REM ENTRY 024
4088  GOSUB ' 102 ("Symbol Set Creation","ATH306. ","6000")        : REM LOAD
4092  DEFFN ' 006                                                  : REM ENTRY 006
4096  GOSUB ' 102 ("SSet:Symbol List","ATH306.0","7000")           : REM LOAD
4100  DEFFN ' 007                                                  : REM ENTRY 007
4104  GOSUB ' 102 ("SSet:Note Diff. <1>","ATH306.1","7000")        : REM LOAD
4108  DEFFN ' 008                                                  : REM ENTRY 008
4112  GOSUB ' 102 ("SSet:Note Diff. <2>","ATH306.2","7000")        : REM LOAD
4116  DEFFN ' 009                                                  : REM ENTRY 009
4120  GOSUB ' 102 ("SSet:Learn Symbol <1>","ATH306.3","7000")      : REM LOAD
4124  DEFFN ' 010                                                  : REM ENTRY 010
4128  GOSUB ' 102 ("SSet:Learn Symbol <2>","ATH306.4","7000")      : REM LOAD
4132  DEFFN ' 011                                                  : REM ENTRY 011
4136  GOSUB ' 102 ("SSet:Practice Symbls <1>","ATH306.5","7000"): REM LOAD
4140  DEFFN ' 012                                                  : REM ENTRY 012
4144  GOSUB ' 102 ("SSet:Practice Symbls <2>","ATH306.6","7000"): REM LOAD
4148  DEFFN ' 013                                                  : REM ENTRY 013
4152  GOSUB ' 102 ("SSet:Symbol Test <1>","ATH306.7","7000")       : REM LOAD
4156  DEFFN ' 014                                                  : REM ENTRY 014
4160  GOSUB ' 102 ("SSet:Symbol Test <2>","ATH306.8","7000")       : REM LOAD
4164  DEFFN ' 015                                                  : REM ENTRY 015
4168  GOSUB ' 102 ("SSet:Symbol Test <3>","ATH306.9","7000")       : REM LOAD
4172  DEFFN ' 025                                                  : REM ENTRY 025
4176  GOSUB ' 102 ("Practice All Symbols <1>","ATH307.0","6000"): REM LOAD
4180  DEFFN ' 026                                                  : REM ENTRY 026
4184  GOSUB ' 102 ("Practice All Symbols <2>","ATH307.1","6000"): REM LOAD
4188  DEFFN ' 027                                                  : REM ENTRY 027
4192  GOSUB ' 102 ("Practice All Symbols <3>","ATH307.2","6000"): REM LOAD
4196  DEFFN ' 028                                                  : REM ENTRY 028
4200  GOSUB ' 102 ("Criterion Test <1>","ATH308.0","6000")         : REM LOAD
4204  DEFFN ' 029                                                  : REM ENTRY 029
4208  GOSUB ' 102 ("Criterion Test <2>","ATH308.1","6000")         : REM LOAD
4212  DEFFN ' 030                                                  : REM ENTRY 030
4216  GOSUB ' 102 ("Suggestions for Training","ATH309.0","6000"): REM LOAD
4220  DEFFN ' 031                                                  : REM ENTRY 031
4224  GOSUB ' 102 ("Main Program <RESTART>","AUTHOR/3","0100")     : REM LOAD
```

```
4228  REM **
4232  REM **
4236  REM **  Subroutine  ********************************************** 2 0 0
4240  REM **  Abstract >
4244  REM **    To generate a random linear subscript.
4248  REM **    Features >
4252  REM **       - Subscripts previously generated will
4256  REM **         be ingored.
4260  REM **       - When 75% of the subscripts have been
4264  REM **         generated the remaining subscripts will
4268  REM **         be returned.
4272  REM **
4276  REM **  Variables >
4280  REM **    B1 > total # of symbls  ! B2 > random number
4284  REM **    B0 > scratch index      ! B8 > random subscript
4288  REM **    B1$> temp byte storage  ! B9 > siz of mem table
4292  REM **    B1$() > memory table    !
4296  REM **
4300  REM ***********************************************
4304  DEFFN ' 200   (B1,B1$(1),B8,B9)                        : REM ENTRY
4308  IF            VAL(B1$(1)) < INT(.75*B1)-1THEN 4320:  REM GENRT RND SBSCRT
4312  GOSUB ' 201   (B1$(1),B8)                              : REM LOCATE NXT SBSCRT
4316  GOTO          4340                                     : REM BRANCH TO EXIT
4320                B2 = INT(100*RND(1)) + 1                 : REM GENRT RND NUMBR
4324                B8 = B2 - INT((B2-1)/B1)*B1              : REM SCALE RND NUMBR
4328  IF            B8 = 0                        THEN 4320  : REM INVALD RESULT
4332  GOSUB ' 202   (B1$,B1$(1),B8)                          : REM UPDATE MEM TBL
4336  IF            B1$ <> HEX(00)                THEN 4320  : REM GENRT NEW RND #
4340                BIN(B1$(1)) = VAL(B1$(1)) + 1            : REM INCR SBL CNT
4344  RETURN                                                 : REM EXIT
```

```
4348  REM **
4352  REM **
4356  REM **  Subroutine  ********************************************** 2 0 1
4360  REM **  Abstract >
4364  REM **    To locate a subscript not previously generated.
4368  REM **  Variables >
4372  REM **    B1$() > memory table       ! B0 > scratch indx
4376  REM **    R5$() > search ids         ! B4 > bit position
4380  REM **    B1$   > temp byte storage  ! B9 > mem tbl size
4384  REM **
4388  REM ************************************************
4392  DEFFN ' 201   (B1$(1),B8)                                  : REM ENTRY
4396  MAT SEARCH    B1$()<2,B9-1>, <>R5$(2) TO R2$()             : REM SRCH MEM TBL
4400                B0 = FNR(VAL(R2$(1))) + 1                    : REM CMPUT TBL INDX
4404                B4 = 1                                       : REM SET BIT INDX
4408  GOSUB ' 203   (B1$(B0),B4)                                 : REM MASK BYTE
4412  IF            B1$ = HEX(00)              THEN 4424         : REM SUBSCRT FOUND
4416                B4 = B4 + 1                                  : REM INCR BIT INDX
4420  GOTO          4408                                         : REM MASK NXT BIT
4424  GOSUB ' 204   (B1$(B0),B4)                                 : REM SET BIT TO ONE
4428                B1$(B0) = B1$                                : REM STOR NEW BIT PTRN
4432                B8 = (B0-2)*8 + B4                           : REM COMPT SUBSCRPT
4436  RETURN                                                     : REM EXIT
```

```
4440  REM **
4444  REM **
4448  REM **  Subroutine  ********************************************** 2 0 2
4452  REM **  Abstract >
4456  REM **   To set a bit on corsponding to the item number.
4460  REM **   If bit is off return a zero bit patern.
4464  REM **  Variables >
4468  REM **    B1$ > temp storage byte ! B2 > item number
4472  REM **    B1$() > memory table    ! B0 > scratch indx
4476  REM **    B4 > bit position       !
4480  REM **
4484  REM ***********************************************
4488  DEFFN ' 202  (B1$,B1$(1),B2)                          : REM ENTRY
4492               B0 = INT((B2-1)/8) + 2                   : REM CMPUT ELEMNT
4496               B4 = B2 - (B0-2)*8                       : REM CMPUT BIT POS
4500  GOSUB ' 203  (B1$(B0),B4)                             : REM MASK BYTE
4504  IF           B1$ <> HEX(00)              THEN 4520    : REM BIT WAS ON
4508  GOSUB ' 204  (B1$(B0),B4)                             : REM TURN BIT ON
4512               B1$(B0) = B1$                            : REM STOR NEW BIT PTRN
4516               B1$ = HEX(00)                            : REM RESET FLAG
4520  RETURN                                                : REM EXIT
```

```
4524  REM **
4528  REM **
4532  REM **  Subroutine  ********************************************** 2 0 3
4536  REM **  Abstract >
4540  REM **    To mask a given bit position.
4544  REM **  Variables >
4548  REM **    B1$ > storage byte   ! B4 > bit position
4552  REM **
4556  REM ********************************************
4560  DEFFN ' 203  (B1$,B4)                                  : REM ENTRY
4564  REM **
4568  REM **       ! BIT 80 ! BIT 40 ! BIT 20 ! BIT 10  !
4572  REM **       !--------!--------!--------!---------!
4576  ON B4 GOTO     4592  ,  4596  ,  4600  ,  4604   ,
                     4608  ,  4612  ,  4616  ,  4620   : REM BRANCH
4580  REM **       !--------!--------!--------!---------!
4584  REM **       ! BIT 08 ! BIT 04 ! BIT 02 ! BIT 01  !
4588  REM **
4592  AND          (B1$,80)                 : GOTO 4624 : REM MASK BIT 8
4596  AND          (B1$,40)                 : GOTO 4624 : REM MASK BIT 7
4600  AND          (B1$,20)                 : GOTO 4624 : REM MASK BIT 6
4604  AND          (B1$,10)                 : GOTO 4624 : REM MASK BIT 5
4608  AND          (B1$,08)                 : GOTO 4624 : REM MASK BIT 4
4612  AND          (B1$,04)                 : GOTO 4624 : REM MASK BIT 3
4616  AND          (B1$,02)                 : GOTO 4624 : REM MASK BIT 2
4620  AND          (B1$,01)                 : GOTO 4624 : REM MASK BIT 1
4624  RETURN                                             : REM EXIT
```

```
4628  REM **
4632  REM **
4636  REM **  Subroutine  ********************************************** 204
4640  REM **  Abstract >
4644  REM **    To turn on a given bit position.
4648  REM **  Variables >
4652  REM **    B1$ > storage byte    ! B4 > bit position
4656  REM **
4660  REM ********************************************
4664  DEFFN ' 204  (B1$,B4)                                  : REM ENTRY
4668  REM **
4672  REM **       ! BIT 80 ! BIT 40 ! BIT 20 ! BIT 10  !
4676  REM **       !--------!--------!--------!---------!
4680  ON B4 GOTO     4696  ,  4700  ,  4704  ,  4708   ,
                     4712  ,  4716  ,  4720  ,  4724   : REM BRANCH
4684  REM **       !--------!--------!--------!---------!
4688  REM **       ! BIT 08 ! BIT 04 ! BIT 02 ! BIT 01  !
4692  REM **
4696  OR           (B1$,80)                : GOTO 4728 : REM MASK BIT 8
4700  OR           (B1$,40)                : GOTO 4728 : REM MASK BIT 7
4704  OR           (B1$,20)                : GOTO 4728 : REM MASK BIT 6
4708  OR           (B1$,10)                : GOTO 4728 : REM MASK BIT 5
4712  OR           (B1$,08)                : GOTO 4728 : REM MASK BIT 4
4716  OR           (B1$,04)                : GOTO 4728 : REM MASK BIT 3
4720  OR           (B1$,02)                : GOTO 4728 : REM MASK BIT 2
4724  OR           (B1$,01)                : GOTO 4728 : REM MASK BIT 1
4728  RETURN                                           : REM EXIT
```

```
4732  REM **
4736  REM **
4740  REM **  Subroutine ********************************************** 2 0 5
4744  REM **  Abstract >
4748  REM **    To Initialize output page buffer from the
4752  REM **    Data Subprogram.
4756  REM **  Variables >
4760  REM **    CO > initial buf indx  ! Q2$() > page buffer
4764  REM **    QO() > prog registr stk!
4768  REM **
4772  REM ***********************************************
4776  DEFFN ' 205  (CO)                                    : REM ENTRY
4780  REM
4784  REM         ! CLR BUFR  !  LOAD BEG  !  SKP FMT    !
4788  REM         ! LOAD BEG  !  PRINT TXT !  NULL       !
4792  REM         ! PRINT TXT !  NULL      !  NULL       !
4796  REM         !-----------!------------!-------------!
4800  ON CO GOTO     4824     ,   4828     ,   4840      : REM BRANCH
4804  REM         !-----------!------------!-------------!
4808  REM
4812  REM **
4816  REM ***********************************************  PROCESS FUNCTIONS
4820  REM **
4824  INIT          (20) Q2$()                             : REM INITIAL BUFR
4828                QO(9) = 1                              : REM SET TO BUFR BEG
4832                QO(10) = 1024                          : REM SET BUFR SIZE
4836  $GIO          ERASE CRT/005(4003,R8$(9))             : REM CLEAR CRT SCRN
4840  GOSUB ' 206  (CO)                                    : REM LOAD BUFR
4844  RETURN                                               : REM EXIT
```

```
4848  REM **
4852  REM **
4856  REM **  Subroutine  ********************************************** 2 0 6
4860  REM **  Abstract >
4864  REM **    To Read the Data Subprogram and load into
4868  REM **    Storage buffer.
4872  REM **  Variables >
4876  REM **    C0 > initial bufr indx  ! C1 > row number
4880  REM **    C2 > col number         ! C3 > length of text
4884  REM **    C4 > control index      ! S4$() > input bufr
4888  REM **
4892  REM ***********************************************
4896  DEFFN ' 206  (C0)                                    : REM ENTRY
4900  IF           C0 <> 3                      THEN 4916 : REM LOAD BUFR
4904  READ         C1,C2,C3,C4,S4$(1)                     : REM READ DATA PROGRAM
4908  IF           C4 <> 9                      THEN 4904 : REM NOT EOF
4912  GOTO         4932                                   : REM BRACNH TO EOF
4916  READ         C1,C2,C3,C4,S4$(1)                     : REM READ DATA PROGRAM
4920  REM **
4924  REM **      ! LOWR ! CAPS ! FONT ! LINE ! UNDRSCR !
4928  REM **      !------!------!------!------!--------!
4932  ON C4 GOTO    4964 , 4984 , 5004 , 5048 ,  5068   ,
                    5104 , 5104 , 5104 , 5124 ,  5104   : REM BRANCH
4936  REM **      !------!------!------!------!--------!
4940  REM **      ! NULL ! NULL ! NULL ! EOF  ! NULL    !
4944  REM **
4948  GOTO        5104                                   : REM INSERT TEXT
4952  REM **
4956  REM ********************************************** CONV TO LOWER CASE
4960  REM **
4964  $TRAN       (S4$() <1,C3>, R4$() <65,64>) R        : REM CONVERSION
4968  GOTO        5104                                   : REM INSERT TEXT
4972  REM **
4976  REM ********************************************** CONV TO UPPER CASE
4980  REM **
4984  $TRAN       (S4$() <1,C3>, R4$() <1,64>)  R        : REM CONVERSION
4988  GOTO        5104                                   : REM INSERT TEXT
4992  REM **
4996  REM ********************************************** ASSIGN FONT TYPE
5000  REM **
5004              C0 = LEN(S4$(1))                       : REM CMPUT LENGTH
5008  MAT COPY    -S4$() <1,C0> TO -S4$()<1,64>          : REM
5012  MAT SEARCH  S4$() <1,64>, <> R5$(1) TO R2$()       : REM
5016              C5 = FNR(VAL(R2$(1)))                  : REM
5020  MAT COPY    R5$() <2,1> TO S4$() <1,1>             : REM
5024  MAT COPY    S4$() <C5,C0> TO S4$() <2,63>          : REM
5028              C3 = C3 + 1                            : REM
5032  GOTO        5104                                   : REM INSERT TEXT
5036  REM **
5040  REM ********************************************** CREATE A LINE
5044  REM **
5048  INIT        (5F) S4$()                             : REM FILL BUFR
5052  GOTO        5104                                   : REM INSERT TEXT
```

```
5056  REM **
5060  REM ********************************************** UNDERSCORE TEXT
5064  REM **
5068                C4 = (C1-1)*64 + C2                  : REM CMPUTE BUF POS
5072  MAT COPY      Q2$() <C4,C3> TO S4$() <1,C3>        : REM LOAD FRM BUFR
5076  AND           (S4$(1),7F)                          : REM MASK TEXT
5080  ADD           (S4$(1),80)                          : REM UNDERSCORE
5084  MAT COPY      S4$() <1,C3>  TO Q2$() <C4,C3>       : REM RELOAD BUFR
5088  GOTO          5104                                 : REM GET NXT LINE
5092  REM **
5096  REM ********************************************** LOAD INTO BUFFER
5100  REM **
5104  GOSUB ' 100   (C1,C2,C3,S4$(1),Q0(12))             : REM INSERT TEXT
5108  GOTO          4916                                 : REM GET NXT LINE
5112  REM **
5116  REM ********************************************** END OF FILE
5120  REM **
5124  GOSUB ' 101   (1,1,0,0)                            : REM HOME CUSOR
5128  RETURN                                             : REM EXIT
```

AUTHOR System: Phase 3 - ATH3AUTO - # MAP

```
4320  - 4308 4328 4336
4340  - 4316
4408  - 4420
4424  - 4412
4520  - 4504
4592  - 4576
4596  - 4576
4600  - 4576
4604  - 4576
4608  - 4576
4612  - 4576
4616  - 4576
4620  - 4576
4624  - 4592 4596 4600 4604 4608 4612 4616 4620
4696  - 4680
4700  - 4680
4704  - 4680
4708  - 4680
4712  - 4680
4716  - 4680
4720  - 4680
4724  - 4680
4728  - 4696 4700 4704 4708 4712 4716 4720 4724
4824  - 4800
4828  - 4800
4840  - 4800
4904  - 4908
4916  - 4900 5108
4932  - 4912
4964  - 4932
4984  - 4932
5004  - 4932
5048  - 4932
5068  - 4932
5104  - 4932 4948 4968 4988 5032 5052 5088
5124  - 4932
```

AUTHOR System: Phase 3 - ATHBAUTO - V MAP

```
B0     - 4400 4408 4424 4428 4432 4492 4496 4500 4508 4512
B1     - 4304 4308 4324
B1$    - 4332 4336 4412 4428 4488 4504 4512 4516 4560 4592 4596 4600 4604 4608
         4612 4616 4620 4664 4696 4700 4704 4708 4712 4716 4720 4724
B1$(   - 4304 4308 4312 4332 4340 4392 4396 4408 4424 4428 4488 4500 4508 4512
B2     - 4320 4324 4488 4492 4496
B4     - 4404 4408 4416 4424 4432 4496 4500 4508 4560 4576 4664 4680
B8     - 4304 4312 4324 4328 4332 4392 4432
B9     - 4304 4396
C0     - 4776 4800 4840 4896 4900 5004 5008 5024
C1     - 4904 4916 5068 5104
C2     - 4904 4916 5068 5104
C3     - 4904 4916 4964 4984 5028 5072 5084 5104
C4     - 4904 4908 4916 4932 5068 5072 5084
C5     - 5016 5024
Q0(    - 4828 4832 5104
Q2$(   - 4824 5072 5084
R2$(   - 4396 4400 5012 5016
R4$(   - 4964 4984
R5$(   - 4396 5012 5020
R8$(   - 4836
S4$(   - 4904 4916 4964 4984 5004 5008 5012 5020 5024 5048 5072 5076 5080 5084
.        5104
```

```
AUTHOR System: Phase 3 - ATH3AUTO - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
4092 DEFFN' 6
4100 DEFFN' 7
4108 DEFFN' 8
4116 DEFFN' 9
4124 DEFFN' 10
4132 DEFFN' 11
4140 DEFFN' 12
4148 DEFFN' 13
4156 DEFFN' 14
4164 DEFFN' 15
4028 DEFFN' 17
4036 DEFFN' 18
4044 DEFFN' 19
4052 DEFFN' 20
4060 DEFFN' 21
4068 DEFFN' 22
4076 DEFFN' 23
4084 DEFFN' 24
4172 DEFFN' 25
4180 DEFFN' 26
4188 DEFFN' 27
4196 DEFFN' 28
4204 DEFFN' 29
4212 DEFFN' 30
4220 DEFFN' 31
???? DEFFN' 100
       - 5104
???? DEFFN' 101
       - 5124
???? DEFFN' 102
       - 4032 4040 4048 4056 4064 4072 4080 4088 4096 4104 4112 4120 4128 4136
         4144 4152 4160 4168 4176 4184 4192 4200 4208 4216 4224
4304 DEFFN' 200
4392 DEFFN' 201
       - 4312
4488 DEFFN' 202
       - 4332
4560 DEFFN' 203
       - 4408 4500
4664 DEFFN' 204
       - 4424 4508
4776 DEFFN' 205
4896 DEFFN' 206
       - 4840
```

5.8 ATH3ØØ.Ø

The purpose of the 'ATH3ØØ.Ø' program is to create the Title page of the learning module.

The remainder of this section presents the 'ATH3ØØ.Ø' program listing and reference maps.

```
6000  %## Program > AUTHOR/3  (ATH300.0)  Version 2.0 2200 T/VP  780322
6010  REM **  Abstract  ************************************************************
6020  REM **                                                                      **
6030  REM **    To Create and Store the Title Page.                               **
6040  REM **                                                                      **
6050  REM ************************************************************************
6060  DEFFN ' 050                                          : REM ENTRY
6070  INIT(20)     Q2$()                                   : REM CLR PAGE BUFR
6080               D3    = 32                              : REM CLR PAGE INDX
6090               Q4    = 00                              : REM CLR ADRS INDX
6100  GOSUB ' 210                                          : REM SAVE LOWER TEXT
6110  GOSUB ' 211                                          : REM CREATE TITLE
6120  GOSUB ' 212                                          : REM CREATE SUBTITLE
6130  GOSUB ' 213                                          : REM CREATE AUTHORS
6140  GOSUB ' 214                                          : REM CREATE DATE
6150  GOSUB ' 215                                          : REM CREATE ORGANIZATION
6160  GOSUB ' 216                                          : REM SAVE PAGE
6170  GOSUB ' 018                                          : REM RETURN TO AUTO MODE
```

```
6180  REM **
6190  REM **
6200  REM **  Subroutine  ********************************************** 2 1 0
6210  REM **  Abstract >
6220  REM **    To save text to be stored on line 16.
6230  REM **  Variables >
6240  REM **    B1$() > temp storage bufr !
6250  REM ***********************************************
6260  DEFFN ' 210                                                  : REM ENTRY
6270  DIM           B0$(1)64                                       : REM DEFINE BUFFER
6280  PRINT         HEX(03)                                        : REM CLR CRT SCREEN
6290                STR(B0$(1),53,11) = "Go to :nxt0"              : REM STOR TEXT
6300  RETURN                                                       : REM EXIT
```

```
6310  REM **
6320  REM **
6330  REM **  Subroutine  ********************************************** 2 1 1
6340  REM **  Abstract >
6350  REM **    To create the TITLE.
6360  REM **  Variables >
6370  REM **    RO$ > begining sctr adrs ! RO > total bytes
6380  REM **
6390  REM ***********************************************
6400  DEFFN ' 211                                          : REM ENTRY
6410  GOSUB ' 105  (RO$,RO,00001)                          : REM FETCH TITLE
6420  GOSUB ' 107  (1,6,60,RO$,0)                          : REM STOR ON PAG
6430  RETURN                                               : REM EXIT
```

```
6440  REM **
6450  REM **
6460  REM **  Subroutine  ********************************************** 2 1 2
6470  REM **  Abstract >
6480  REM **    To create the SUBTITLE.
6490  REM **  Variables >
6500  REM **    RO$ > begining sctr adrs ! RO > total bytes
6510  REM **
6520  REM ***********************************************
6530  DEFFN ' 212                                          : REM ENTRY
6540  GOSUB ' 105  (RO$,RO,00002)                          : REM FETCH SUBTITLE
6550  GOSUB ' 107  (Q1+2,6,60,RO$,0)                       : REM STOR ON PAG
6560  RETURN                                               : REM EXIT
```

```
6570  REM **
6580  REM **
6590  REM **  Subroutine  ********************************************** 2 1 3
6600  REM **  Abstract >
6610  REM **    To create the AUTHORS.
6620  REM **  Variables >
6630  REM **    RO$ > begining sctr adrs ! RO > total bytes
6640  REM **
6650  REM ********************************************
6660  DEFFN ' 213                                          : REM ENTRY
6670  GOSUB ' 105  (RO$,RO,00003)                          : REM FETCH AUTHORS
6680  GOSUB ' 107  (Q1+2,6,60,RO$,0)                       : REM STOR ON PAC
6690  RETURN                                               : REM EXIT
```

```
6700  REM **
6710  REM **
6720  REM **  Subroutine  ********************************************** 2 1 4
6730  REM **  Abstract >
6740  REM **    To create the DATE.
6750  REM **  Variables >
6760  REM **    RO$ > begining sctr adrs ! RO > total bytes
6770  REM **
6780  REM ************************************************
6790  DEFFN ' 214                                          : REM ENTRY
6800  GOSUB ' 105  (RO$,RO,00004)                          : REM FETCH AUTHORS
6810  GOSUB ' 107  (Q1+2,6,60,RO$,0)                       : REM STOR ON PAG
6820  RETURN                                               : REM EXIT
```

```
6830  REM **
6840  REM **
6850  REM **  Subroutine  ********************************************** 2 1 5
6860  REM **  Abstract >
6870  REM **    To create the NAME OF THE ORGANIZATION.
6880  REM **  Variables >
6890  REM **    RO$ > begining sctr adrs ! RO > total bytes
6900  REM **
6910  REM ***********************************************
6920  DEFFN ' 215                                          : REM ENTRY
6930  GOSUB ' 105  (RO$,RO,00005)                          : REM FETCH AUTHORS
6940  GOSUB ' 107  (Q1+2,6,60,RO$,0)                       : REM STOR ON PAG
6950  RETURN                                               : REM EXIT
```

```
6960  REM **
6970  REM **
6980  REM **  Subroutine  ********************************************** 2 1 6
6990  REM **  Abstract >
7000  REM **    To Save current page on output disk.
7010  REM **  Variables >
7020  REM **    B1$() > temp storage buffer !
7030  REM **
7040  REM ***********************************************
7050  DEFFN ' 216                                          : REM ENTRY
7060  GOSUB ' 100  (16,1,64,B0$(1),0)                      : REM STOR LOWER TEXT
7070  GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7080  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH300.0 - # MAP

AUTHOR System: Phase 3 - ATH300.0 - V MAP

```
B0$(   - 6270 6290 7060
D1     - 7070
D2     - 7070
D3     - 6080 7070
D4$    - 7070
D5     - 7070
Q1     - 6550 6680 6810 6940
Q2$(   - 6070
Q4     - 6090
R0     - 6410 6540 6670 6800 6930
R0$    - 6410 6420 6540 6550 6670 6680 6800 6810 6930 6940
```

AUTHOR System: Phase 3 - ATH300.0 - ' MAP

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 18
       - 6170
6060 DEFFN' 50
???? DEFFN' 100
       - 7060
???? DEFFN' 105
       - 6410 6540 6670 6800 6930
???? DEFFN' 107
       - 6420 6550 6680 6810 6940
???? DEFFN' 110
       - 7070
6260 DEFFN' 210
       - 6100
6400 DEFFN' 211
       - 6110
6530 DEFFN' 212
       - 6120
6660 DEFFN' 213
       - 6130
6790 DEFFN' 214
       - 6140
6920 DEFFN' 215
       - 6150
7050 DEFFN' 216
       - 6160
```

5.9 ATH3Ø1.Ø

The purpose of the 'ATH3Ø1.Ø' program is to create the Introduction page of the learning module.

The remainder of this section presents the 'ATH3Ø1.Ø' program listing and reference maps.

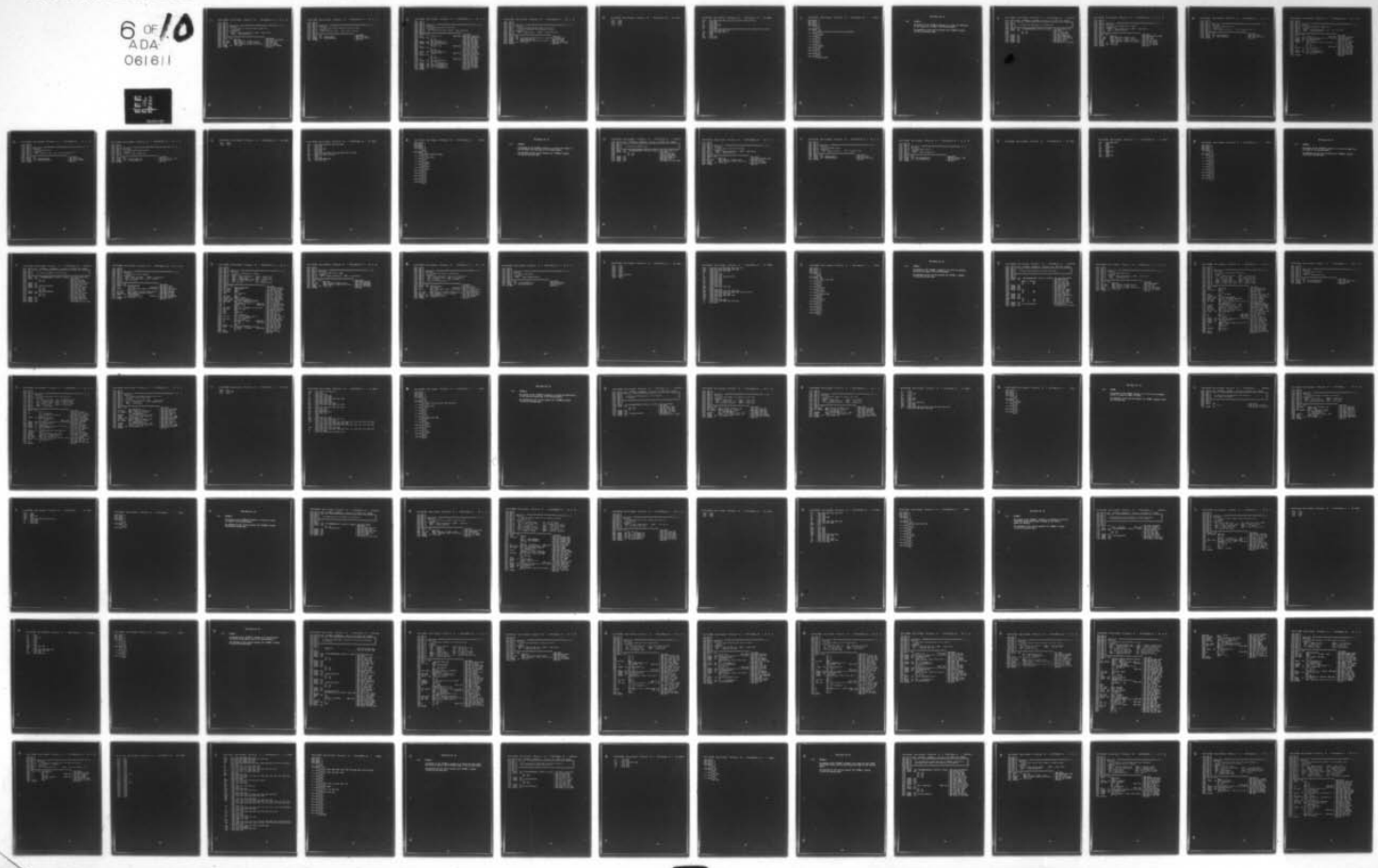
AUTHOR System: Phase 3 - ATH301.0 - CNTRL

```
6000  %## Program > AUTHOR/3  (ATH301.0)  Version > 2.0 2200 T/VP  780322
6010  REM **  Abstract  ************************************************************
6020  REM **                                                                       **
6030  REM **  To create and store the Introduction Page                            **
6040  REM **                                                                       **
6050  REM ***************************************************************************
6060  GOSUB ' 102 ("DATA SUBPROGRAM","ATH101.0","9000"): REM LOAD FIXED TEXT
6070  DEFFN ' 050                                      : REM ENTRY
6080              D3 = 32                              : REM CLR PAG INDX
6090              Q4 = 41                              : REM SET ADRS INDX
6100  GOSUB ' 210                                      : REM CLR LOWER TEXT
6110  GOSUB ' 211                                      : REM LOAD INTRO TEXT
6120  GOSUB ' 212                                      : REM RE-LOAD LOWR TEXT
6130  GOSUB ' 110 (D1,D2,D3,D4$,D5)                    : REM SAVE PAGE
6140  GOSUB ' 019                                      : REM RETURN TO AUTO MODE
```

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U)
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

6 OF 10
ADA
061611



```
6150  REM **
6160  REM **
6170  REM **  Subroutine  ********************************************** 2 1 0
6180  REM **  Abstract >
6190  REM **    To save test on lower portion of page then
6200  REM **    clear page.
6210  REM **  Variables >
6220  REM **    B0$() > temp storage bufr ! Q2$() > page buffer
6230  REM **    R4$() > conversion bufr   !
6240  REM **
6250  REM ***********************************************
6260  DEFFN ' 210                                                : REM ENTRY
6270  DIM           B0$(9)64                                     : REM DEFINE STOR BUFR
6280  MAT COPY      Q2$()<385,512> TO B0$()<1,512>               : REM STOR TEXT
6290  MAT COPY      Q2$()<961,64>  TO B0$()<513,64>              : REM STOR TEXT
6300  $TRAN         (Q2$()<385,640>, R4$()<177,4>)00 R           : REM CLR BUFFER
6310  GOSUB ' 101   (7,1,10,0)                                   : REM CLR CRT SCREEN
6320  RETURN                                                     : REM EXIT
```

```
6330  REM **
6340  REM **
6350  REM **  Subroutine  ********************************************** 2 1 1
6360  REM **  Abstract >
6370  REM **    To load the training objective onto the page.
6380  REM **  Variables >
6390  REM **    RO$ > item sctr adrs   ! RO > bufr rtrn size
6400  REM **
6410  REM ********************************************
6420  DEFFN ' 211                                          : REM ENTRY
6430  GOSUB ' 105  (RO$,RO,00011)                          : REM FETCH TEXT
6440  GOSUB ' 107  (2,24,60,RO$,0)                         : REM STOR ON PAGE
6450  RETURN                                               : REM EXIT
```

```
6460  REM **
6470  REM **
6480  REM **  Subroutine  ********************************************** 2 1 2
6490  REM **  Abstract >
6500  REM **    To re-load page with fixed text.
6510  REM **  Variables >
6520  REM **    Q1 > current row indx ! BO$() > temp storg bufr
6530  REM **
6540  REM ***********************************************
6550  DEFFN ' 212                                          : REM ENTRY
6560  IF          16-Q1-1 >= 3              THEN 6590      : REM SPACE AVAIABLE
6570  GOSUB ' 213                                          : REM SAVE CURRNT PAG
6580              Q1 = 0                                   : REM SET ROW INDX
6590              Q1 = Q1 + 2                              : REM INCR ROW INDX
6600  GOSUB ' 100 (Q1,1,64,BO$(1),0)                       : REM LOAD PAG BUFR
6610  GOSUB ' 100 (Q1+1,1,64,BO$(2),0)                     : REM LOAD PAG BUFR
6620              Q1 = Q1 + 1                              : REM INCR ROW INDX
6630  IF          16-Q1-1 >= 2              THEN 6660      : REM SPACE AVAIABLE
6640  GOSUB ' 213                                          : REM SAVE CURRNT PAG
6650              Q1 = 0                                   : REM SET ROW INDX
6660              Q1 = Q1 + 2                              : REM INCR ROW INDX
6670  GOSUB ' 100 (Q1,1,64,BO$(4),0)                       : REM LOAD PAG BUFR
6680  IF          16-Q1-1 >= 4              THEN 6710      : REM SPACE AVAIABLE
6690  GOSUB ' 213                                          : REM SAVE CURRNT PAG
6700              Q1 = 1                                   : REM SET ROW INDX
6710              Q1 = Q1 + 1                              : REM INCR ROW INDX
6720  IF          Q1 = 2                    THEN 6750      : REM NO LINE NEEDED
6730  GOSUB ' 100 (Q1,1,64,BO$(5),0)                       : REM LOAD PAG BUFR
6740              Q1 = Q1 + 1                              : REM INCR ROW INDX
6750  GOSUB ' 100 (Q1  ,1,64,BO$(6),0)                     : REM LOAD PAG BUFR
6760  GOSUB ' 100 (Q1+1,1,64,BO$(7),0)                     : REM LOAD PAG BUFR
6770  GOSUB ' 100 (Q1+2,1,64,BO$(8),0)                     : REM LOAD PAG BUFR
6780  GOSUB ' 100 (16  ,1,64,BO$(9),0)                     : REM LOAD PAG BUFR
6790  RETURN                                               : REM EXIT
```

```
6800  REM **
6810  REM **
6820  REM **  Subroutine  ********************************************** 2 1 3
6830  REM **  Abstract >
6840  REM **    To save current page and clear in order to
6850  REM **    allow the loading of the remainder of text.
6860  REM **  Variables >
6870  REM **    Q2$() > page buffer    ! R4$() > conversion bufr
6880  REM **
6890  REM ********************************************
6900  DEFFN ' 213                                          : REM ENTRY
6910  GOSUB ' 100  (16,41,23,"To Continue Go To :nxt0",0): REM CONT MSG
6920  GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
6930  $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R       : REM CLR BUFFR
6940  GOSUB ' 101  (2,1,15,0)                              : REM CLR CRT SCREEN
6950  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH301.0 - # MAP

6590 - 6560
6660 - 6630
6710 - 6680
6750 - 6720

AUTHOR System: Phase 3 - ATH301.O - V MAP

```
B0$(   - 6270 6280 6290 6600 6610 6670 6730 6750 6760 6770 6780
D1     - 6130 6920
D2     - 6130 6920
D3     - 6080 6130 6920
D4$    - 6130 6920
D5     - 6130 6920
Q1     - 6560 6580 6590 6600 6610 6620 6630 6650 6660 6670 6680 6700 6710 6720
         6730 6740 6750 6760 6770
Q2$(   - 6280 6290 6300 6930
Q4     - 6090
R0     - 6430
R0$    - 6430 6440
R4$(   - 6300 6930
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 19
       - 6140
6070 DEFFN' 50
???? DEFFN' 100
       - 6600 6610 6670 6730 6750 6760 6770 6780 6910
???? DEFFN' 101
       - 6310 6940
???? DEFFN' 102
       - 6060
???? DEFFN' 105
       - 6430
???? DEFFN' 107
       - 6440
???? DEFFN' 110
       - 6130 6920
6260 DEFFN' 210
       - 6100
6420 DEFFN' 211
       - 6110
6550 DEFFN' 212
       - 6120
6900 DEFFN' 213
       - 6570 6640 6690
```

5.1Ø ATH3Ø2.Ø

The purpose of the 'ATH3Ø2.Ø' program is to create the Importance of this Program to You page of the learning module.

The remainder of this section presents the 'ATH3Ø2.Ø' program listing and reference maps.

```
6000  %## Program > AUTHOR/III  (ATH302.0)  Version > 2.0 2200 T/VP  780322
6010  REM **  Abstract  ************************************************************
6020  REM **                                                                       **
6030  REM **    To create the Importance of Program page.                          **
6040  REM **                                                                       **
6050  REM ************************************************************************
6060  GOSUB ' 102  ("DATASUBPROGRAM","ATH102.0","9000"): REM LOAD FORMAT TEXT
6070  DEFFN ' 050                                      : REM ENTRY
6080                 D3 = 32                           : REM CLR PAG INDX
6090                 Q4 = 00                           : REM CLR ADRS INDX
6100  GOSUB ' 210                                      : REM SAVE LOWER TEXT
6110  GOSUB ' 211                                      : REM WRITE IMPORTANCE
6120  GOSUB ' 212                                      : REM WRITE LOWR HEADING
6130  GOSUB ' 213                                      : REM WRITE USAGE
6140  GOSUB ' 214                                      : REM SAVE PAGE
6150  GOSUB ' 020                                      : REM RETURN TO AUTO MODE
```

```
6160  REM **
6170  REM **
6180  REM **  Subroutine  ********************************************  2 1 0
6190  REM **  Abstract >
6200  REM **    To save lower heading of the page.
6210  REM **  Variables >
6220  REM **    BO$() > temp storage bufr ! Q2$() > page buffer
6230  REM **    R4$() > conversion bufr   !
6240  REM **
6250  REM ********************************************
6260  DEFFN ' 210                                          : REM ENTRY
6270  DIM           BO$(4)64                               : REM DEFINE STORAGE BUFR
6280  MAT COPY      Q2$()<449,192> TO BO$()<1,192>         : REM STOR TEXT
6290  MAT COPY      Q2$()<961,64>  TO BO$()<193,64>        : REM STOR TEXT
6300  $TRAN         (Q2$()<449,192>, R4$()<177,4>)00 R     : REM CLR PAG BUFFR
6310  $TRAN         (Q2$()<961,64> , R4$()<177,4>)00 R     : REM CLR PAG BUFFR
6320  GOSUB ' 101   (8,1,3,0)                              : REM CLR CRT SCREEN
6330  GOSUB ' 101   (16,1,1,0)                             : REM CLR CRT SCREEN
6340  RETURN                                               : REM EXIT
```

```
6350  REM **
6360  REM **
6370  REM **  Subroutine  ********************************************  2 1 1
6380  REM **  Abstract >
6390  REM **    To write Importance text.
6400  REM **  Variables >
6410  REM **    RO$ > begin sctr adrs     ! RO > length of bufr
6420  REM **
6430  REM ***********************************************
6440  DEFFN ' 211                                          : REM ENTRY
6450  GOSUB ' 105  (RO$,RO,00031)                          : REM FETCH TEXT
6460  GOSUB ' 107  (2,24,60,RO$,0)                         : REM LOAD ONTO PAGE
6470  RETURN                                               : REM EXIT
```

```
6480  REM **
6490  REM **
6500  REM **  Subroutine  ********************************************  2 1 2
6510  REM **  Abstract >
6520  REM **    To Write Heading for Usage Text.
6530  REM **  Variables >
6540  REM **    BO$() > temp storage bufr ! Q1 > curnt row indx
6550  REM **    R4$() > conversion bufr   !
6560  REM **
6570  REM ***********************************************
6580  DEFFN ' 212                                                  : REM ENTRY
6590  IF          16-Q1-1 >= 4                        THEN 6650  : REM SPACE AVAIABLE
6600  GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):REM CUNTINUE MSG
6610  GOSUB ' 110  (D1,D2,D3,D4$,D5)                           : REM SAVE PAGE
6620  GOSUB ' 101  (2,1,15,0)                                  : REM CLR CRT SCREEN
6630  $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R           : REM CLR PAG BUFR
6640               Q1 = 1                                      : REM SET ROW INDX
6650               Q1 = Q1 + 1                                 : REM INCR ROW INDX
6660  IF           Q1 = 2                              THEN 6690 : REM SKIP UNDERLINE
6670  GOSUB ' 100  (Q1,1,64,BO$(1),0)                          : REM LOAD PAG BUFR
6680               Q1 = Q1 + 1                                 : REM INCR ROW INDX
6690  GOSUB ' 100  (Q1  ,1,64,BO$(2),0)                        : REM LOAD PAG BUFR
6700  GOSUB ' 100  (Q1+1,1,64,BO$(3),0)                        : REM LOAD PAG BUFR
6710               Q1 = Q1 + 2                                 : REM INCR ROW INDX
6720  RETURN                                                   : REM EXIT
```

```
6730  REM **
6740  REM **
6750  REM **  Subroutine  ********************************************** 2 1 3
6760  REM **  Abstract >
6770  REM **    To Write Usage text.
6780  REM **  Variables >
6790  REM **    RO$ > item sctr adrs ! RO > siz of storage bufr
6800  REM **
6810  REM ***********************************************
6820  DEFFN ' 213                                          : REM ENTRY
6830  GOSUB ' 105  (RO$,RO,00032)                          : REM FETCH TEXT
6840  GOSUB ' 107  (Q1,24,60,RO$,0)                        : REM STOR IN BUFFR
6850  RETURN                                               : REM EXIT
```

```
6860  REM **
6870  REM **
6880  REM **  Subroutine  ********************************************** 2 1 4
6890  REM **  Abstract >
6900  REM **    To save current page on disk.
6910  REM **  Variables >
6920  REM **    BO$() > temp storage bufr !
6930  REM **
6940  REM ********************************************
6950  DEFFN ' 214                                          : REM ENTRY
6960  GOSUB ' 100  (16,1,64,BO$(4),0)                      : REM RE-LOAD LST LINE
6970  GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
6980  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH302.0 - # MAP

6650 - 6590
6690 - 6660

AUTHOR System: Phase 3 - ATH302.O - V MAP

```
B0$(   - 6270 6280 6290 6670 6690 6700 6960
D1     - 6610 6970
D2     - 6610 6970
D3     - 6080 6610 6970
D4$    - 6610 6970
D5     - 6610 6970
Q1     - 6590 6640 6650 6660 6670 6680 6690 6700 6710 6840
Q2$(   - 6280 6290 6300 6310 6630
Q4     - 6090
R0     - 6450 6830
R0$    - 6450 6460 6830 6840
R4$(   - 6300 6310 6630
```

```
AUTHOR System: Phase 3 - ATH302.0 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 20
       - 6150
6070 DEFFN' 50
???? DEFFN' 100
       - 6600 6670 6690 6700 6960
???? DEFFN' 101
       - 6320 6330 6620
???? DEFFN' 102
       - 6060
???? DEFFN' 105
       - 6450 6830
???? DEFFN' 107
       - 6460 6840
???? DEFFN' 110
       - 6610 6970
6260 DEFFN' 210
       - 6100
6440 DEFFN' 211
       - 6110
6580 DEFFN' 212
       - 6120
6820 DEFFN' 213
       - 6130
6950 DEFFN' 214
       - 6140
```

5.11 ATH3Ø3.Ø

The purpose of the 'ATH3Ø3.Ø' program is to create the Example of How the Symbols Are Used page of the learning module.

The remainder of this section presents the 'ATH3Ø3.Ø' program listing and reference maps.

```
6000  %## Program > AUTHOR/III  (ATH303.0)  Version > 2.0 2200 T/VP  780322
6010  REM **  Abstract  ************************************************************
6020  REM **                                                                       **
6030  REM **    To create the Example Page.                                        **
6040  REM **                                                                       **
6050  REM ***************************************************************************
6060  GOSUB ' 102  ("DATASUBPROGRAM","ATH103.0","9000"): REM LOAD FORMAT TEXT
6070  DEFFN ' 050                                     : REM ENTRY
6080                D3 = 32                           : REM CLR PAG INDX
6090                Q4 = 00                           : REM CLR ADRS INDX
6100  GOSUB ' 210                                     : REM SAVE LOWER TEXT
6110  GOSUB ' 211                                     : REM WRITE EXAMPLE
6120  GOSUB ' 212                                     : REM SAVE PAGE
6130  GOSUB ' 021                                     : REM RETURN TO AUTO MODE
```

```
6140  REM **
6150  REM **
6160  REM **  Subroutine  ********************************************** 2 1 0
6170  REM **  Abstract >
6180  REM **    To save text on last line of page.
6190  REM **  Variables >
6200  REM **    BO$() > temp storage bufr ! Q2$() > page buffer
6210  REM **    R4$() > conversion bufr   !
6220  REM **
6230  REM ********************************************
6240  DEFFN ' 210                                              : REM ENTRY
6250  DIM           BO$(1)64                                   : REM DEFINE STORAGE BUFR
6260  MAT COPY      Q2$()<961,64> TO BO$()<1,64>               : REM STOR TEXT
6270  $TRAN         (Q2$()<961,64>, R4$()<177,4>)00 R          : REM CLR PAG BUFFR
6280  GOSUB ' 101   (16,1,1,0)                                 : REM CLR CRT SCREEN
6290  RETURN                                                   : REM EXIT
```

```
6300  REM **
6310  REM **
6320  REM **  Subroutine  ********************************************** 2 1 1
6330  REM **  Abstract >
6340  REM **    To write Example text.
6350  REM **  Variables >
6360  REM **    RO$ > begin sctr adrs     ! RO > length of bufr
6370  REM **
6380  REM ********************************************
6390  DEFFN ' 211                                          : REM ENTRY
6400  GOSUB ' 105  (RO$,RO,00041)                          : REM FETCH TEXT
6410  GOSUB ' 107  (2,06,60,RO$,0)                         : REM LOAD ONTO PAGE
6420  RETURN                                               : REM EXIT
```

```
6430  REM **
6440  REM **
6450  REM **  Subroutine  ********************************************** 2 1 2
6460  REM **  Abstract >
6470  REM **    To Save page on Disk.
6480  REM **  Variables >
6490  REM **    BO$() > temp storage bufr !
6500  REM **
6510  REM ********************************************
6520  DEFFN ' 212                                          : REM ENTRY
6530  GOSUB ' 100  (16,1,64,BO$(1),0)                      : REM RE-LOAD LST LINE
6540  GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
6550  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH303.0 - # MAI

AUTHOR System: Phase 3 - ATH303.0 - V MAP

```
B0$(   - 6250 6260 6530
D1     - 6540
D2     - 6540
D3     - 6080 6540
D4$    - 6540
D5     - 6540
Q2$(   - 6260 6270
Q4     - 6090
R0     - 6400
R0$    - 6400 6410
R4$(   - 6270
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 21
       - 6130
6070 DEFFN' 50
???? DEFFN' 100
       - 6530
???? DEFFN' 101
       - 6280
???? DEFFN' 102
       - 6060
???? DEFFN' 105
       - 6400
???? DEFFN' 107
       - 6410
???? DEFFN' 110
       - 6540
6240 DEFFN' 210
       - 6100
6390 DEFFN' 211
       - 6110
6520 DEFFN' 212
       - 6120
```

5.12 ATH3Ø4.Ø

The purpose of the 'ATH3Ø4.Ø' program is to create the Symbol Pre-Test pages of the learning module.

The remainder of this section presents the 'ATH3Ø4.Ø' program listing and reference maps.

```
6000  %## Program > AUTHOR/III  (ATH304.0)   Version > 2.0 2200 T/VP  780322
6010  REM **  Abstract  ************************************************************
6020  REM **                                                                       **
6030  REM **    To create the Symbol Pre-Test Frames.                              **
6040  REM **                                                                       **
6050  REM ***************************************************************************
6060  GOSUB ' 102  ("DATASUBPROGRAM","ATH104.0","9000"): REM LOAD FORMAT TEXT
6070  DEFFN ' 050                                      : REM ENTRY
6080                D3 = 32                            : REM CLR PAG INDX
6090                Q4 = 42                            : REM SET ADRS INDX
6100  GOSUB ' 210                                      : REM WRITE TYPE OF SBL
6110  GOSUB ' 211                                      : REM SAMPLES OF SBLS
6120  GOSUB ' 110  (D1,D2,D3,D4$,D5)                   : REM SAVE PAGE
6130  GOSUB ' 205  (1)                                 : REM LOAD NXT FORMAT
6140                D3 = 32                            : REM CLR PAG INDX
6150                Q4 = 00                            : REM CLR ADRS INDX
6160  GOSUB ' 110  (D1,D2,D3,D4$,D5)                   : REM SAVE PAGE
6170  GOSUB ' 205  (1)                                 : REM LOAD NXT FORMAT
6180                D3 = 32                            : REM CLR PAG INDX
6190                Q4 = 00                            : REM CLR ADRS INDX
6200  GOSUB ' 212                                      : REM SAVE LOWR TEXT
6210  GOSUB ' 213                                      : REM WRITE REFERENCE
6220  GOSUB ' 214                                      : REM SAVE PAGE
6230  GOSUB ' 022                                      : REM RETURN TO AUTO MODE
```

```
6240  REM **
6250  REM **
6260  REM **  Subroutine  ********************************************** 2 1 0
6270  REM **  Abstract >
6280  REM **    To Write the Type of Symbol.
6290  REM **  Variables >
6300  REM **    R0$ > begin sctr adrs    ! Q2$() > page buffer
6310  REM **    R1$() > storage buffer   ! Q1 > row indx
6320  REM **    Q2 > col indx            !
6330  REM **
6340  REM ***********************************************
6350  DEFFN ' 210                                                : REM ENTRY
6360  GOSUB ' 105  (R0$,R0,00022)                                : REM FETCH TEXT
6370  IF           R0 <= 60                        THEN 6400     : REM SRING SIZ OK
6380  $TRAN        (R1$()<60,1957>, R4$()<177,4>)00 R            : REM CLR STORAG BUFR
6390               R0 = 60                                       : REM RESET TO MAX
6400  MAT COPY     R1$()<1,R0+9> TO R1$()<R0+10,R0+9>            : REM POSITION BUFR
6410  MAT COPY     Q2$()<408,9> TO R1$()<1,R0+9>                 : REM MOVE IN TEXT
6420  GOSUB ' 107  (7,24,60,R0$,0)                               : REM LOAD PAC BUFR
6430  IF           Q2$(Q1,Q2-2) = "."              THEN 6450     : REM CHCK FOR PERIOD
6440  GOSUB ' 100  (Q1,Q2-1,1,".",0)                             : REM LOAD PERIOD
6450  RETURN                                                     : REM EXIT
```

```
6460  REM **
6470  REM **
6480  REM **  Subroutine  ********************************************** 2 1 1
6490  REM **  Abstract >
6500  REM **    To write a sampleing of symbols.
6510  REM **  Variables >
6520  REM **    B0$() > temp storage bufr ! B1$() > memory bufr
6530  REM **    B2$() > memory bufr       ! Q3$() > symbol tbl
6540  REM **    Q0$() > alpha numeric flgs! G0() > numeric flag
6550  REM **    Q9$() > scratch i/o bufr  ! T1 > row indx
6560  REM **    T2 > col indx             ! B8 > random sbscrpt
6570  REM **
6580  REM ***********************************************
6590  DEFFN ' 211                                          : REM ENTRY
6600  DIM           B0$(1)64,B2$(20)1                      : REM DEFINE BUFFERS
6610  MAT REDIM     Q9$(16)16                              : REM DEFINE I/O BUFR
6620  INIT(00)      B1$()                                  : REM INIT MEM BUFR
6630  INIT(00)      B2$()                                  : REM INIT MEM BUFR
6640  FOR           T1 = 12 TO 15                          : REM BUILD FOUR ROWS
6650                T2 = 24                                : REM INIT COL INDX
6660  INIT(20)      B0$()                                  : REM INIT TEMP BUFR
6670  MAT COPY      B2$() TO B1$()                         : REM UPDAT MEM BUFR
6680  GOSUB ' 200   (G0(2),B1$(1),B8,20)                   : REM FTCH RND SBSCRT
6690  DATA LOAD     BA T #2, (Q3$(B8,3),R0$)Q9$()          : REM FTCH SYMBOL
6700  MAT COPY      R5$()<1,1> TO Q9$()<16,1>              : REM CLR EOI BYTE
6710  IF            T2+LEN(Q9$(1)) < 60       THEN 6790    : REM LIN NOT EXCEEDED
6720  IF            Q0$(3) <> "3"             THEN 6770    : REM FIND SHORTER SBL
6730                Q0$(3) =  "0"                          : REM RE-SET FLAG
6740  MAT COPY      R5$()<1,1> TO B0$()<T2-1,64-T2+2>      : REM CLR END OF BUFR
6750  MAT COPY      B2$() TO B1$()                         : REM REFRESH MEM BUFR
6760  GOTO          6870                                   : REM BRNCH TO PRNT
6770  ADD           (Q0$(3),01)                            : REM INCR FLAG
6780  GOTO          6670                                   : REM FTCH NXT SBL
6790                T3 = LEN(Q9$(1))                       : REM STOR SBL LEN
6800  MAT COPY      Q9$()<1,T3> TO B0$()<T2,T3>            : REM STOR SBL IN BUFR
6810                T2 = T2 + LEN(Q9$(1))                  : REM INCR COL INDX
6820  MAT COPY      B1$() TO B2$()                         : REM UPDATE MEM BUFR
6830  IF            VAL(B1$(1)) = G0(2)       THEN 6870    : REM SBL TBL EMPTY
6840  IF            T2 > 56                   THEN 6870    : REM LIN BUFR FUL
6850                T2 = T2 + 3                            : REM INCR COL INDX
6860  GOTO          6680                                   : REM BRNCH TO STRT
6870  GOSUB ' 100   (T1,24,40,STR(B0$(1),24,40),0)         : REM LOAD PAC BUFR
6880  IF            VAL(B1$(1)) <> G0(2)      THEN 6900    : REM FTCH NXT SBL
6890                T1 = 15                                : REM EXIT LOOP
6900  NEXT          T1                                     : REM CONT TO LOAD
6910  RETURN                                               : REM EXIT
```

```
6920  REM **
6930  REM **
6940  REM **  Subroutine  ********************************************** 2 1 2
6950  REM **  Abstract >
6960  REM **    To save lower part of page.
6970  REM **  Variables >
6980  REM **    BO$() > temp storage bufr ! Q2$() > page buffer
6990  REM **
7000  REM ********************************************
7010  DEFFN ' 212                                          : REM ENTRY
7020  DIM           BO$(1)64                               : REM DEFINE BUFFER
7030  MAT COPY      Q2$()<961,64> TO BO$()<1,64>           : REM LOAD TEMP BUFR
7040  $TRAN         (Q2$()<961,64>, R4$()<177,4>)00 R      : REM CLR PAG BUFFR
7050  GOSUB ' 101   (16,1,1,0)                             : REM CLR CRT SCREEN
7060  RETURN                                               : REM EXIT
```

```
7070  REM **
7080  REM **
7090  REM **  Subroutine  ********************************************** 2 1 3
7100  REM **  Abstract >
7110  REM **    To write Offical Source of Reference.
7120  REM **  Variables >
7130  REM **    RO$ > begin sctr adrs ! RO > siz of text bufr
7140  REM **    R1$() > storage buffr ! Q2$() > pag buffer
7150  REM **    Q1 > row indx         ! Q2 > col indx
7160  REM **    R4$() > conversion buf!
7170  REM **
7180  REM ***********************************************
7190  DEFFN ' 213                                          : REM ENTRY
7200  GOSUB ' 105  (RO$,RO,00023)                          : REM FETCH TEXT
7210  IF           RO <= 160                    THEN 7240  : REM BUF SIZ OK
7220  $TRAN        (R1$()<160,1856>, R4$()<177,4>)00 R     : REM CLR STORAGE BUFR
7230               RO = 160                                : REM SET TO MAXIUM
7240  MAT COPY     R1$()<1,RO+21> TO R1$()<RO+22,RO+21>:  REM POSITION BUFR TEXT
7250  MAT COPY     Q2$()<600,21> TO R1$()<1,RO+21>         : REM LOAD FIXED TEXT
7260  GOSUB ' 107  (10,24,60,RO$,0)                        : REM LOAD PAG BUFFER
7270  GOSUB ' 100  (Q1,Q2-1,2,").",0)                      : REM LOAD PERIOD
7280  RETURN                                               : REM EXIT
```

```
7290  REM **
7300  REM **
7310  REM **  Subroutine  ********************************************** 2 1 4
7320  REM **  Abstract >
7330  REM **    To save current Page.
7340  REM **  Variables >
7350  REM **    BO$() > temp storage buffer !
7360  REM **
7370  REM ********************************************
7380  DEFFN ' 214                                          : REM ENTRY
7390  GOSUB ' 100  (16,1,64,BO$(1),0)                      : REM RE-LOAD TEXT
7400  GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7410  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH304.0 - # MAP

```
6400  - 6370
6450  - 6430
6670  - 6780
6680  - 6860
6770  - 6720
6790  - 6710
6870  - 6760 6830 6840
6900  - 6880
7240  - 7210
```

AUTHOR System: Phase 3 - ATH304.O - V MAP

```
B0$(   - 6600 6660 6740 6800 6870 7020 7030 7390
B1$(   - 6620 6670 6680 6750 6820 6830 6880
B2$(   - 6600 6630 6670 6750 6820
B8     - 6680 6690
D1     - 6120 6160 7400
D2     - 6120 6160 7400
D3     - 6080 6120 6140 6160 6180 7400
D4$    - 6120 6160 7400
D5     - 6120 6160 7400
Q0(    - 6680 6830 6880
Q0$(   - 6720 6730 6770
Q1     - 6430 6440 7270
Q2     - 6430 6440 7270
Q2$(   - 6410 6430 7030 7040 7250
Q3$(   - 6690
Q4     - 6090 6150 6190
Q9$(   - 6610 6690 6700 6710 6790 6800 6810
R0     - 6360 6370 6390 6400 6410 7200 7210 7230 7240 7250
R0$    - 6360 6420 6690 7200 7260
R1$(   - 6380 6400 6410 7220 7240 7250
R4$(   - 6380 7040 7220
R5$(   - 6700 6740
T1     - 6640 6870 6890 6900
T2     - 6650 6710 6740 6800 6810 6840 6850
T3     - 6790 6800
```

AUTHOR System: Phase 3 - ATH304.0 - ' MAP

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 22
       - 6230
6070 DEFFN' 50
???? DEFFN' 100
       - 6440 6870 7270 7390
???? DEFFN' 101
       - 7050
???? DEFFN' 102
       - 6060
???? DEFFN' 105
       - 6360 7200
???? DEFFN' 107
       - 6420 7260
???? DEFFN' 110
       - 6120 6160 7400
???? DEFFN' 200
       - 6680
???? DEFFN' 205
       - 6130 6170
6350 DEFFN' 210
       - 6100
6590 DEFFN' 211
       - 6110
7010 DEFFN' 212
       - 6200
7190 DEFFN' 213
       - 6210
7380 DEFFN' 214
       - 6220
```

5.13 ATH3Ø4.1

The purpose of the 'ATH3Ø4.1' program is to create the Optional Criterion Test pages of the learning module.

The remainder of this section presents the 'ATH3Ø4.1' program listing and reference maps.

```
6000  %## Program > AUTHOR/3  (ATH304.1)   Version > 2.0  2200 T/VP  780322
6010  REM **  Abstract  ********************************************************
6020  REM **                                                                    **
6030  REM **    To Create the Optional Criterion Test Frames                    **
6040  REM **                                                                    **
6050  REM ************************************************************************
6060  REM **
6070  REM **
6080  GOSUB ' 102  ("DATASUBPROGRAM","ATH104.1","9000"): REM LOAD PAGE FORMAT
6090  DEFFN ' 050                                      : REM ENTRY POINT
6100                Q0$(4) = HEX(31)                   : REM SET COL FLAG
6110                D3     =     32                    : REM CLR PAGE INDX
6120                Q4     =     00                    : REM CLR ADRS INDX
6130  GOSUB ' 210                                      : REM SAVE LOWR TEXT
6140  GOSUB ' 211                                      : REM CREATE TEST
6150  GOSUB ' 212                                      : REM SAVE PAGE
6160  GOSUB ' 205  (1)                                 : REM LOAD NXT FORMAT
6170                D3     =     00                    : REM SET PAGE INDX
6180                Q4     =     00                    : REM CLR ADRS INDX
6190  GOSUB ' 210                                      : REM SAVE LOWR TEXT
6200  GOSUB ' 213                                      : REM CREATE ANSWER PAG
6210  GOSUB ' 212                                      : REM SAVE PAGE
6220  GOSUB ' 205  (1)                                 : REM LOAD NXT FORMAT
6230                D3     =     32                    : REM CLR PAGE INDX
6240                Q4     =     00                    : REM CLR ADRS INDX
6250  GOSUB ' 214                                      : REM COMPUTE LOW SCORE
6260  GOSUB ' 110  (D1,D2,D3,D4$,D5)                   : REM SAVE PAGE
6270  GOSUB ' 023                                      : REM RETURN TO AUTO MODE
```

```
6280  REM **
6290  REM **
6300  REM **  Subroutine  ********************************************** 2 1 0
6310  REM **  Abstract >
6320  REM **    To save lower page text.
6330  REM **  Variables >
6340  REM **    B0$() > temp storage bufr ! Q2$() > page buffr
6350  REM **    R4$() > conversion bufr   !
6360  REM **
6370  REM ***********************************************
6380  DEFFN ' 210                                          : REM ENTRY
6390  DIM           B0$(2)64                               : REM DEFINE BUFFER
6400  MAT COPY      Q2$()<321,64> TO B0$()<65,64>          : REM STOR TEXT
6410  MAT COPY      Q2$()<961,64> TO B0$()<1,64>           : REM STOR TEXT
6420  $TRAN         (Q2$()<961,64>, R4$()<177,4>)00 R      : REM CLR PAGE BUFFER
6430  GOSUB ' 101   (16,1,1,0)                             : REM CLR CRT SCREEN
6440  RETURN                                               : REM EXIT
```

```
6450  REM **
6460  REM **
6470  REM **  Subroutine  ********************************************** 2 1 1
6480  REM **  Abstract >
6490  REM **    To create criterion Test Questions.
6500  REM **  Variables >
6510  REM **    T1$() > memory sbl ordr  ! Q9$() > scratch bufr
6520  REM **    T1 > total sbl indx      ! T2 > curnt row indx
6530  REM **    T3 > curnt col indx      ! B8 > random subscrpt
6540  REM **    Q3$() > symbol table     ! R9$() > temp lin buf
6550  REM **    R5$() > single char bufr ! R4$() > conversions
6560  REM **    Q0$() > single byte flgs ! Q0() > numeric flags
6570  REM **
6580  REM ************************************************
6590  DEFFN ' 211                                          : REM ENTRY
6600  DIM           T1$(75)16                              : REM DEFINE BUFFER
6610  INIT(00)      B1$()                                  : REM ZERO MEM TABLE
6620                T2 = 2                                 : REM ROW INDX
6630                T3 = 6                                 : REM COL INDX
6640  FOR           T1 = 1 TO Q0(2)                        : REM BEGIN SBL TBL LOOP
6650  INIT(SF)      R9$()                                  : REM INIT LINE BUFR
6660  GOSUB ' 200   (Q0(2),B1$(1),B8,20)                   : REM FETCH RNDM SBSCRPT
6670  DATA LOAD     BA T #2, (Q3$(B8,3),R0$)Q9$()          : REM LOAD SYMBOL
6680  MAT COPY      R5$()<1,1> TO Q9$()<16,1>              : REM CLR EOI CHAR
6690  MAT COPY      Q9$()<1,15> TO T1$()<(T1-1)*16+1,16>   : REM SAVE SYMBOL
6700                T4 = LEN(T1$(T1))                      : REM CMPUT SBL LEN
6710                BIN(STR(T1$(T1),16,1)) = B8            : REM REMEMBER SEQUENCE
6720  CONVERT       T1 TO STR(R9$(1),1,2), (##)            : REM CMPUTE ITEM NUMBR
6730  $TRAN         (R9$()<1,1>, R4$()<185,4>) R           : REM BLANK FILL NUMBR
6740  MAT COPY      R5$()<8,1> TO R9$()<3,2>               : REM LOAD DECIMAL POINT
6750  MAT COPY      Q9$()<1,T4> TO R9$()<5,T4+2>           : REM LOAD SYMBOL
6760  GOSUB ' 100   (T2,T3,25,R9$(1),0)                    : REM LOAD INTO PAC BUFR
6770                T2 = T2 + 1                            : REM INCR ROW INDX
6780  IF            T2 < 16                  THEN 6900     : REM NOT END OF PAC
6790  IF            Q0$(4) <> "2"            THEN 6860     : REM NOT COL 2
6800  IF            T1 = Q0(2)               THEN 6900     : REM END OF SBL TBL
6810  GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0)    : REM LOAD CONT MSG
6820  GOSUB ' 110   (D1,D2,D3,D4$,D5)                      : REM SAVE CURRNT PAGE
6830  GOSUB ' 101   (2,1,15,0)                             : REM CLR CRT SCREEN
6840  $TRAN         (Q2$()<65,960>, R4$()<177,4>)00 R      : REM CLR PAGE BUFR
6850                Q0$(4) = "0"                           : REM RESET COL INCTOR
6860  ADD           (Q0$(4),01)                            : REM INCR COL INCTOR
6870                T2 = 2                                 : REM RESET ROW INDX
6880  CONVERT       Q0$(4) TO T4                           : REM CONV COL NUMBR
6890                T3 = 6+14                              : REM CMPUTE COL INDX
6900  NEXT          T1                                     : REM CONT SBL TBL LOOP
6910  RETURN                                               : REM EXIT
```

```
6920  REM **
6930  REM **
6940  REM **  Subroutine  ********************************************** 2 1 2
6950  REM **  Abstract  >
6960  REM **    To save page on disk.
6970  REM **  Variables >
6980  REM **    B0$() > temp storage bufr !
6990  REM **
7000  REM ************************************************
7010  DEFFN ' 212                                          : REM ENTRY
7020  GOSUB ' 100  (16,1,64,B0$(1),0)                      : REM LOAD SAVED TEXT
7030  GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7040  RETURN                                               : REM EXIT
```

```
7050  REM **
7060  REM **
7070  REM **  Subroutine  ********************************************** 2 1 3
7080  REM **  Abstract >
7090  REM **    To create the Answers for Criterion Test.
7100  REM **  Variables >
7110  REM **    T2 > curnt row indx  ! T1 > sbl sequence indx
7120  REM **    T4 > symbol tbl indx ! T1$() > seqnce mem tbl
7130  REM **    Q3$() > symbol table ! Q0() > numeric flags
7140  REM **    R0 > length of ans.  ! R9$() > temp line bufr
7150  REM **    Q9$() > i/o buffer   ! R4$() > conversions
7160  REM **    R5$() > single chars !
7170  REM **
7180  REM ************************************************
7190  DEFFN ' 213                                          : REM ENTRY
7200                T2 = 7                                 : REM SET ROW INDX
7210  FOR           T1 = 1 TO Q0(2)                        : REM BEG SBL TBL LOOP
7220                T4 = VAL(STR(T1$(T1),16,1))            : REM CMPUT SBL TBL INDX
7230                Q0$(1) = "1"                           : REM SET NULL LOOKUP
7240  GOSUB ' 105   (Q3$(T4,4),R0,R1)                      : REM LOAD DEFINITION
7250                R0 = INT(R0/32) + 1                    : REM CMPUT LINE NEEDED
7260  IF            R0 <= 16-T2                THEN 7330 : REM SPACE AVAIABLE
7270  GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):  REM LOAD CONT MSG
7280  GOSUB ' 110   (D1,D2,D3,D4$,D5)                      : REM SAVE PAGE
7290  GOSUB ' 101   (2,1,15,0)                             : REM CLR CRT SCREEN
7300  $TRAN         (Q2$()<65,960>, R4$()<177,4>)00 R      : REM CLR PAGE BUFR
7310  GOSUB ' 100   (2,1,64,B0$(2),0)                      : REM LOAD COL HEADING
7320                T2 = 3                                 : REM SET ROW INDX
7330  GOSUB ' 107   (T2,26,60,Q3$(T4,4),VAL(Q0$(10)))      : REM LOAD TEXT
7340  INIT(20)      R9$()                                  : REM CLR LINE BUFR
7350  CONVERT       T1 TO STR(R9$(1),1,2), (##)            : REM CMPUT ITEM NUMBR
7360  $TRAN         (R9$()<1,1>, R4$()<185,4>) R           : REM ZERO FILL NUMBR
7370  MAT COPY      R5$()<8,1> TO R9$()<3,2>               : REM LOAD DECIMAL POINT
7380  MAT COPY      T1$()<(T1-1)*16+1,15> TO R9$()<5,16>:  REM LOAD SYMBOL
7390  GOSUB ' 100   (T2,6,20,R9$(1),0)                     : REM LOAD INTO PAG BUFR
7400                T2 = Q1 + 1                            : REM INCR ROW INDX
7410  NEXT          T1                                     : REM CONT SBL TBL LOOP
7420  RETURN                                               : REM EXIT
```

```
7430  REM **
7440  REM **
7450  REM **  Subroutine  ********************************************** 2 1 4
7460  REM **  Abstract >
7470  REM **    To calculate the lowest passing score and
7480  REM **    to insert into the page text.
7490  REM **  Variables >
7500  REM **    T1 > pos to insrt score! Q2$() > page buffer
7510  REM **    R9$() > temp buffer     ! T2 > lowest score
7520  REM **    R4$() > conversions     !
7530  REM **
7540  REM ***********************************************
7550  DEFFN ' 214                                          : REM ENTRY
7560                T2 = Q0(2)*.20                         : REM CMPUT LOW SCORE
7570  MAT COPY      Q2$()<65,64> TO R9$()<1,64>            : REM LOAD BUFR TEXT
7580  MAT SEARCH    R9$(), =R5$(10) TO R2$()               : REM SRCH FOR KEY
7590                T1 = FNR(VAL(R2$(1)))                  : REM CMPUT POSITION
7600  CONVERT       T2 TO STR(R9$(1),T1,2), (##)           : REM CONV SCORE
7610  $TRAN         (R9$()<T1,1>, R4$()<185,4>) R          : REM BLANK FILL NUMBR
7620  GOSUB ' 100   (2,1,64,R9$(1),0)                      : REM RELOAD PAG BUFR
7630  MAT COPY      Q2$()<449,64> TO R9$()<1,64>           : REM LOAD BUFR TEXT
7640  MAT SEARCH    R9$(), =R5$(10) TO R2$()               : REM SRCH FOR KEY
7650                T1 = FNR(VAL(R2$(1)))                  : REM CMPUT POSITION
7660  CONVERT       T2 TO STR(R9$(1),T1,2), (##)           : REM CONV SCORE
7670  $TRAN         (R9$()<T1,1>, R4$()<185,4>) R          : REM BLANK FILL NUMBR
7680  GOSUB ' 100   (8,1,64,R9$(1),0)                      : REM RE-LOAD BUFFER
7690  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH304.1 - # MAP

6860 - 6790
6900 - 6780 6800
7330 - 7260

```
B0$(   - 6390 6400 6410 7020 7310
B1$(   - 6610 6660
B8     - 6660 6670 6710
D1     - 6260 6820 7030 7280
D2     - 6260 6820 7030 7280
D3     - 6110 6170 6230 6260 6820 7030 7280
D4$    - 6260 6820 7030 7280
D5     - 6260 6820 7030 7280
Q0(    - 6640 6660 6800 7210 7560
Q0$(   - 6100 6790 6850 6860 6880 7230 7330
Q1     - 7400
Q2$(   - 6400 6410 6420 6840 7300 7570 7630
Q3$(   - 6670 7240 7330
Q4     - 6120 6180 6240
Q9$(   - 6670 6680 6690 6750
R0     - 7240 7250 7260
R0$    - 6670
R1     - 7240
R2$(   - 7580 7590 7640 7650
R4$(   - 6420 6730 6840 7300 7360 7610 7670
R5$(   - 6680 6740 7370 7580 7640
R9$(   - 6650 6720 6730 6740 6750 6760 7340 7350 7360 7370 7380 7390 7570 7580
         7600 7610 7620 7630 7640 7660 7670 7680
T1     - 6640 6690 6700 6710 6720 6800 6900 7210 7220 7350 7380 7410 7590 7600
         7610 7650 7660 7670
T1$(   - 6600 6690 6700 6710 7220 7380
T2     - 6620 6760 6770 6780 6870 7200 7260 7320 7330 7390 7400 7560 7600 7660
T3     - 6630 6760 6890
T4     - 6700 6750 6880 6890 7220 7240 7330
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 23
       - 6270
6090 DEFFN' 50
???? DEFFN' 100
       - 6760 6810 7020 7270 7310 7390 7620 7680
???? DEFFN' 101
       - 6430 6830 7290
???? DEFFN' 102
       - 6080
???? DEFFN' 105
       - 7240
???? DEFFN' 107
       - 7330
???? DEFFN' 110
       - 6260 6820 7030 7280
???? DEFFN' 200
       - 6660
???? DEFFN' 205
       - 6160 6220
6380 DEFFN' 210
       - 6130 6190
6590 DEFFN' 211
       - 6140
7010 DEFFN' 212
       - 6150 6210
7190 DEFFN' 213
       - 6200
7550 DEFFN' 214
       - 6250
```

5.14 ATH3Ø5.Ø

The purpose of the 'ATH3Ø5.Ø' program is to create the Organization of the Learning Program page of the learning module.

The remainder of this section presents the 'ATH3Ø5.Ø' program listing and reference maps.

```
6000  %## Program > AUTHOR/3  (ATH305.0)   Version > 2.0 2200 1/VP  780322
6010  REM **  Abstract  ***************************************************
6020  REM **                                                               **
6030  REM **    To Create the Organization of the Program.                 **
6040  REM **    (Flowchart)                                                **
6050  REM **                                                               **
6060  REM *******************************************************************
6070  REM **
6080  REM **
6090  GOSUB ' 102("DATASUBPROGRAM","ATH105.0","9000"): REM LOAD FORMAT
6100  DEFFN ' 050                                    : REM ENTRY
6110              D3 = 00                            : REM SET PAGE INDX
6120              Q4 = 43                            : REM SET ADRS INDX
6130  GOSUB ' 210                                    : REM INSRT TOT SETS
6140  GOSUB ' 211                                    : REM INSRT TOT SBLS
6150  GOSUB ' 110  (D1,D2,D3,D4$,D5)                 : REM SAVE PAGE
6160  GOSUB ' 024                                    : REM RETURN TO AUTO MODE
```

```
6170  REM **
6180  REM **
6190  REM **  Subroutine  ********************************************** 2 1 0
6200  REM **  Abstract >
6210  REM **    To insert the number of symbol sets into text
6220  REM **  Variables >
6230  REM **    Q2$() > page buffer       ! R9$() > line buffer
6240  REM **    R5$() > single char bufr  ! R2$() > rutrn srch
6250  REM **    T1 > position of insrt    ! R4$() > conversion
6260  REM **
6270  REM ***********************************************
6280  DEFFN ' 210                                          : REM ENTRY
6290  MAT COPY      Q2$()<129,64> TO R9$()<1,64>           : REM LOAD LINE BUFR
6300  MAT SEARCH    R9$(), = R5$(10) TO R2$()              : REM SRCH LINE BUFR
6310                T1 = FNR(VAL(R2$(1)))                  : REM CMPUT POSITION
6320  CONVERT       Q0(1) TO STR(R9$(1),T1,2), (##)        : REM CONV SBL SETS #
6330  $TRAN         (R9$()<T1,1>, R4$()<185,4>) R          : REM BLANK FILL NUMBR
6340  GOSUB ' 100   (3,1,64,R9$(1),0)                      : REM LOAD INTO PAG BUFR
6350  RETURN                                               : REM EXIT
```

```
6360  REM **
6370  REM **
6380  REM **  Subroutine  ********************************************  2 1 1
6390  REM **  Abstract >
6400  REM **    To insert the number of symbols into text.
6410  REM **  Variables >
6420  REM **    Q2$() > page buffer       ! R9$() > line buffer
6430  REM **    R5$() > single char bufr  ! R2$() > rutrn srch
6440  REM **    T1 > position of insrt    ! R4$() > conversion
6450  REM **
6460  REM ************************************************
6470  DEFFN ' 211                                          : REM ENTRY
6480  MAT COPY      Q2$()<321,64> TO R9$()<1,64>           : REM LOAD LINE BUFR
6490  MAT SEARCH    R9$(), = R5$(10) TO R2$()              : REM SRCH LINE BUFR
6500                T1 = FNR(VAL(R2$(1)))                  : REM CMPUT POSITION
6510  CONVERT       QO(2) TO STR(R9$(1),T1,2), (##)        : REM CONV SBL SETS #
6520  $TRAN         (R9$()<T1,1>, R4$()<185,4>) R          : REM BLANK FILL NUMBR
6530  GOSUB ' 100   (6,1,64,R9$(1),0)                      : REM LOAD INTO PAC BUFR
6540  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH305.0 - V MAP

```
D1     - 6150
D2     - 6150
D3     - 6110 6150
D4$    - 6150
D5     - 6150
Q0(    - 6320 6510
Q2$(   - 6290 6480
Q4     - 6120
R2$(   - 6300 6310 6490 6500
R4$(   - 6330 6520
R5$(   - 6300 6490
R9$(   - 6290 6300 6320 6330 6340 6480 6490 6510 6520 6530
T1     - 6310 6320 6330 6500 6510 6520
```

```
AUTHOR System: Phase 3 - ATH305.0 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 24
       - 6160
6100 DEFFN' 50
???? DEFFN' 100
       - 6340 6530
???? DEFFN' 102
       - 6090
???? DEFFN' 110
       - 6150
6280 DEFFN' 210
       - 6130
6470 DEFFN' 211
       - 6140
```

5.15 ATH3Ø6.

The purpose of the 'ATH3Ø6' program is to initialize the programs needed to create the Symbol Set pages.

The remainder of this section presents the 'ATH3Ø6' program listing and reference maps.

```
6000  %## Program > AUTHOR/3  (ATH306. )  Version > 2.0 2200 1/VP  780322
6010  REM **  Abstract  ************************************************************
6020  REM **                                                                       **
6030  REM **    To Initialize the generation and storage of                        **
6040  REM **    the symbol set Module.                                             **
6050  REM **                                                                       **
6060  REM ***************************************************************************
6070  REM **
6080  REM **
6090  DEFFN ' 060                                          : REM ENTRY
6100              U0 = 1                                   : REM SET SBL TBL INDX
6110  GOSUB ' 006                                          : REM RETURN TO LOAD SEQU
```

```
6120  REM **
6130  REM **
6140  REM **  Subroutine  ********************************************** 2 1 0
6150  REM **  Abstract >
6160  REM **    To load the symbol set number in heading.
6170  REM **  Variables >
6180  REM **    Q7$() > sbl id tbl       ! Q2$() > page buffer
6190  REM **    R9$() > temp line bufr   ! R2$() > rutrn srch
6200  REM **    U0 > curnt sbl tbl indx ! T2 > pos to insrt
6210  REM **
6220  REM ***********************************************
6230  DEFFN ' 210                                          : REM ENTRY
6240  MAT COPY      Q2$()<1,64> TO R9$()<1,64>             : REM LOAD LINE BUFR
6250  ADD           (R9$(1),80)                            : REM CLR UNDRSCORE
6260  MAT SEARCH    R9$(), = R5$(10) TO R2$()              : REM SRCH LINE BUFR
6270                T2 = FNR(VAL(R2$(1))) + 1              : REM CMPUT POSITION
6280                T3 = VAL(Q7$(U0,1))                    : REM CMPUT SBL SET #
6290  CONVERT       T3 TO STR(R9$(1),T2,2), (##)           : REM CONV SBL SET #
6300  ADD           (R9$(1),80)                            : REM RESET UNDRSCORE
6310  GOSUB ' 100   (1,1,64,R9$(1),0)                      : REM LOAD PAGE BUFFR
6320  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH306. - V MAP

```
Q2$(   - 6240
Q7$(   - 6280
R2$(   - 6260 6270
R5$(   - 6260
R9$(   - 6240 6250 6260 6290 6300 6310
T2     - 6270 6290
T3     - 6280 6290
U0     - 6100 6280
```

```
AUTHOR System: Phase 3 - ATH306.    -  ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 6
       - 6110
6090 DEFFN' 60
???? DEFFN' 100
       - 6310
6230 DEFFN' 210
```

5.16 ATH3Ø6.Ø

The purpose of the 'ATH3Ø6.Ø' program is to create the symbol list page of the learning module.

The remainder of this section presents the 'ATH3Ø6.Ø' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.0 - CNTRL

```
7000 %## Program > AUTHOR/3  (ATH306.0)   Version > 2.0 2200 T/VP  780322
7010 REM **  Abstract  ************************************************************
7020 REM **                                                                        **
7030 REM **    To Create the List of symbol in current set.                        **
7040 REM **                                                                        **
7050 REM ************************************************************************
7060 REM **
7070 REM **
7080 GOSUB ' 102  ("DATASUBPROGRAM","ATH106.0","9000"): REM LOAD FORMAT
7090 DEFFN.' 050                                      : REM ENTRY
7100         D3 = 32                                  : REM CLR PAGE INDX FLG
7110         Q4 = VAL(Q7$(U0,1))                      : REM SET ADRS INDX
7120 GOSUB ' 210                                      : REM LOAD SBL SET NUMBR
7130 GOSUB ' 211                                      : REM SAVE LOWR TEXT
7140 GOSUB ' 212                                      : REM LOAD SYMBL LIST
7150 GOSUB ' 213                                      : REM SAVE PAGE
7160 GOSUB ' 007                                      : REM RETURN TO LOAD SEQU
```

```
7170 REM **
7180 REM **
7190 REM **  Subroutine  ********************************************** 2 1 1
7200 REM **  Abstract >
7210 REM **    To save the text in lower portion of page.
7220 REM **  Variables >
7230 REM **    BO$() > temp storage bufr ! Q2$() > page bufr
7240 REM **    R4$() > conversion bufr   !
7250 REM **
7260 REM ***********************************************
7270  DEFFN ' 211                                          : REM ENTRY
7280  DIM          BO$(4)64                                : REM DEFINE STOR BUFR
7290  MAT COPY     Q2$()<769,256> TO BO$()<1,256>          : REM LOAD BUFFER
7300  $TRAN        (Q2$()<769,256>, R4$()<177,4>)00 R      : REM CLR TEXT BUFR
7310  GOSUB ' 101 (13,1,4,0)                               : REM CLR CRT SCREEN
7320  RETURN                                               : REM EXIT
```

```
7330 REM **
7340 REM **
7350 REM **  Subroutine  ********************************************** 2 1 2
7360 REM **  Abstract >
7370 REM **    To load text buffer with list of symbols contained
7380 REM **    in current set.
7390 REM **  Variables >
7400 REM **    R1$() > storage buffer      ! U0 > sbl tbl index
7410 REM **    T2 > sbl tbl indx (temp)    ! T3 > storage buffer indx
7420 REM **    Q9$() > i/o buffer          ! T4$() > translation buffer
7430 REM **    Q7$() > symbol id table     ! Q3$() > symbl lokup table
7440 REM **    Q1 > text bufr line indx    ! Q2$() > text buffer
7450 REM **    R4$() > conversion bufr     ! R0$() > scratch reg bufr
7460 REM **    T4 > length of symbol       ! R9$() > scratch i/o buffer
7470 REM **
7480 REM ********************************************
7490  DEFFN ' 212                                          : REM ENTRY
7500  DIM          T4$(2)4                                 : REM DEFINE WORK BUFR
7510  INIT(20)     R1$()                                   : REM CLR STORAGE BUFR
7520               T4$(1) = HEX(09202020)                  : REM DEFN TRANSLATION
7530               T4$(2) = HEX(20092020)                  : REM DEFN TRANSLATION
7540               T2 = U0                                 : REM SET SBL TBL INDX
7550               T3 = 1                                  : REM INIT BUFR INDX
7560  IF           Q7$(U0,1) <> Q7$(T2,1)     THEN 7660    : REM END OF SBL SET
7570  DATA LOAD    BA T #2, (Q3$(T2,3),R0$) Q9$()          : REM LOAD SYMBOL
7580  MAT COPY     Q9$()<1,15> TO R9$()<1,64>              : REM STOR SYMBOL
7590               T4 = LEN(R9$(1))                        : REM CMPUT SBL LENGTH
7600  $TRAN        (R9$()<1,T4>, T4$()<1,4>) R             : REM CONV INTERNL BLNKS
7610               STR(R9$(1),T4+2,3) = HEX(0909)          : REM ALOW FOR TRIPLE SP
7620  MAT COPY     R9$()<1,T4+5> TO R1$()<T3,20>           : REM LOAD STOR BUFR
7630               T2 = T2 + 1                             : REM INCR SBL TBL INDX
7640               T3 = T3 + 20                            : REM INCR BUFR INDX
7650  GOTO         7560                                    : REM CHCK NXT SYMBL
7660  GOSUB ' 107 (4,24,60,R0$,0)                          : REM LOAD SBL LIST
7670  $TRAN        (Q2$(), T4$()<5,4>) R                   : REM CONV INTERNL BLNKS
7680  IF           16-Q1-1 >= 4               THEN 7740    : REM SPACE REMAINING
7690  GOSUB ' 100 (16,41,23,"To Continue Go to :nxt0",0): REM EOP CONT MSG
7700  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAGE ON DSK
7710  GOSUB ' 101 (2,1,15,0)                               : REM CLR CRT SCREEN
7720  $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R       : REM CLR TEXT BUFR
7730               Q1 = 0                                  : REM RESET LINE INDX
7740  RETURN                                               : REM EXIT
```

```
7750 REM **
7760 REM **
7770 REM **  Subroutine  ********************************************** 2 1 3
7780 REM **  Abstract >
7790 REM **    To load fixed text in the remaing space and save
7800 REM **    page on disk.
7810 REM **  Variables >
7820 REM **    BO$() > temp storage buffer  ! Q2$() > text buffer
7830 REM **    Q1 > text bufr line indx     !
7840 REM **
7850 REM ********************************************
7860  DEFFN ' 213                                          : REM ENTRY
7870              Q1 = Q1 + 2                              : REM INCR LINE INDX
7880  GOSUB ' 100 (Q1  ,1,64,BO$(1),0)                     : REM LOAD PAGE BUFR
7890  GOSUB ' 100 (Q1+1,1,64,BO$(2),0)                     : REM LOAD PAGE BUFR
7900  GOSUB ' 100 (Q1+2,1,64,BO$(3),0)                     : REM LOAD PAGE BUFR
7910  GOSUB ' 100 (16  ,1,64,BO$(4),0)                     : REM LOAD PAGE BUFR
7920  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAG ON DSK
7930  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH306.0 - # MAP

7560 - 7650
7660 - 7560
7740 - 7680

```
B0$(   - 7280 7290 7880 7890 7900 7910
D1     - 7700 7920
D2     - 7700 7920
D3     - 7100 7700 7920
D4$    - 7700 7920
D5     - 7700 7920
Q1     - 7680 7730 7870 7880 7890 7900
Q2$(   - 7290 7300 7670 7720
Q3$(   - 7570
Q4     - 7110
Q7$(   - 7110 7560
Q9$(   - 7570 7580
R0$    - 7570 7660
R1$(   - 7510 7620
R4$(   - 7300 7720
R9$(   - 7580 7590 7600 7610 7620
T2     - 7540 7560 7570 7630
T3     - 7550 7620 7640
T4     - 7590 7600 7610 7620
T4$(   - 7500 7520 7530 7600 7670
U0     - 7110 7540 7560
```

```
AUTHOR System:  Phase 3 - ATH306.0 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 7
       - 7160
7090 DEFFN' 50
???? DEFFN' 100
       - 7690 7880 7890 7900 7910
???? DEFFN' 101
       - 7310 7710
???? DEFFN' 102
       - 7080
???? DEFFN' 107
       - 7660
???? DEFFN' 110
       - 7700 7920
???? DEFFN' 210
       - 7120
7270 DEFFN' 211
       - 7130
7490 DEFFN' 212
       - 7140
7860 DEFFN' 213
       - 7150
```

5.17 ATH3Ø6.1

The purpose of the 'ATH3Ø6.1' program is to determine if the Note Differences Among Symbols page is needed for the current symbol set sequence.

The remainder of this section presents the 'ATH3Ø6.1' program listing and reference maps.

```
7000 %## Program > AUTHOR/3  (ATH306.1)   Version > 2.0 2200 T/VP  780322
7010 REM **  Abstract  ************************************************************
7020 REM **                                                                        **
7030 REM **    To determine if it is nessasary to create the frames                **
7040 REM **    covering the Differences between similar symbols                    **
7050 REM **                                                                        **
7060 REM ************************************************************************
7070 REM **
7080 REM **
7090  GOSUB ' 211                                                : REM FIND SIM SYMBLS
7100  IF        U1$(1) = HEX(00)              THEN 7180 : REM NO SIM SYMBLS
7110  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.1","9000"): REM LOAD FORMAT DATA
7120  DEFFN ' 050                                                : REM DEFINE ENTRY
7130            D3 = 32                                          : REM CLR PAGE INDX
7140            Q4 = 00                                          : REM CLR ADRS INDX
7150  GOSUB ' 210                                                : REM LOAD SET NUMBR
7160  GOSUB ' 110 (D1,D2,D3,D4$,D5)                              : REM SAVE PAG ON DSK
7170  GOSUB ' 008                                                : REM RTURN TO LOAD SEGU
7180  GOSUB ' 009                                                : REM RTURN TO AUTO MODE
```

```
7190 REM **
7200 REM **
7210 REM **  Subroutine  ********************************************** 2 1 1
7220 REM **  Abstract >
7230 REM **    To find any similar symbos withen the current set
7240 REM **    of symbols.
7250 REM **  Variables >
7260 REM **    U0 > symbol table indx        ! T2 > symbo table indx (temp)
7270 REM **    T3 > sim sbl tbl indx (temp) ! Q7$() > symbl id table
7280 REM **    Q3$() > symbl lokup table     ! Q9$() > i/o buffer
7290 REM **    R0$() > scratch reg bufr      ! U1$() > sim sbl lokup table
7300 REM **    U1 > sim sbl tbl indx         !
7310 REM **
7320 REM ************************************************
7330  DEFFN ' 211                                          : REM ENTRY
7340  MAT REDIM   Q9$(32)8                                 : REM DEFINE I/O BUFR
7350              T2 = U0                                  : REM INIT SBL TBL INDX
7360              T3 = 1                                   : REM INIT SIM TBL INDX
7370  IF          Q7$(U0,1) <> Q7$(T2,1)    THEN 7440      : REM END OF SBL SET
7380  DATA LOAD   BA T #2, (Q3$(T2,6),R0$) Q9$()           : REM LOAD SIM SBLS
7390  IF          STR(Q9$(1),4,4) = "9999"  THEN 7420      : REM NO SIM SBLS
7400              BIN(U1$(T3)) = T2                        : REM LOAD LOKUP TABLE
7410              T3 = T3 + 1                              : REM INCR SIM TBL INDX
7420              T2 = T2 + 1                              : REM INCR SBL TBL INDX
7430  GOTO        7370                                     : REM CHCK NXT SYMBL
7440              U1$(T3) = HEX(00)                        : REM SET END OF TBL FLG
7450  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH306.1 - # MAP

7180 - 7100
7370 - 7430
7420 - 7390
7440 - 7370

AUTHOR System: Phase 3 - ATH306.1 - V MAP

```
D1      - 7160
D2      - 7160
D3      - 7130 7160
D4$     - 7160
D5      - 7160
Q3$(    - 7380
Q4      - 7140
Q7$(    - 7370
Q9$(    - 7340 7380 7390
R0$     - 7380
T2      - 7350 7370 7380 7400 7420
T3      - 7360 7400 7410 7440
U0      - 7350 7370
U1$(    - 7100 7400 7440
```

AUTHOR System: Phase 3 - ATH306.1 - ' MAP

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 8
       - 7170
???? DEFFN' 9
       - 7180
7120 DEFFN' 50
???? DEFFN' 102
       - 7110
???? DEFFN' 110
       - 7160
???? DEFFN' 210
       - 7150
7330 DEFFN' 211
       - 7090
```

5.18 ATH3Ø6.2

The purpose of the 'ATH3Ø6.2' program is to create the Note Differences Among Symbols page of the learning module.

The remainder of this section presents the 'ATH3Ø6.2' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.2 - CNTRL

```
7000 %## Program > AUTHOR/3  (ATH306.2)   Version > 2.0 2200 T/VP  780322
7005 REM **  Abstract  ************************************************
7010 REM **                                                           **
7015 REM **    To Construct the frames covering the 'Differences between    **
7020 REM **    the Symbols'.                                          **
7025 REM **                                                           **
7030 REM ***************************************************************
7035 REM **
7040 REM **
7045  DIM         U2$(10)16                                  : REM DEFINE WORK BUFR
7050              U1 = 1                                     : REM INIT SIM SBL INDX
7055 REM **
7060 REM **
7065 GOSUB ' 102  ("DATASUBPROGRAM","ATH106.2","9000")       : REM LOAD PAGE FMTS
7070 DEFFN ' 050                                             : REM ENTRY
7075 GOSUB ' 211                                             : REM LOAD SIM TABLE
7080              D3 = 0                                     : REM SET PAGE INDX
7085              Q4 = 00                                    : REM CLR ADRS INDX
7090 GOSUB ' 210                                             : REM INSERT SBL SET #
7095 GOSUB ' 212                                             : REM SAVE LOWER TEXT
7100 GOSUB ' 213                                             : REM INSERT SBL LIST
7105 GOSUB ' 214                                             : REM SAVE PAGE
7110 GOSUB ' 205  (1)                                        : REM LOAD NXT FORMAT
7115              D3 = 32                                    : REM CLR PAGE INDX
7120              Q4 = 00                                    : REM CLR ADRS INDX
7125 GOSUB ' 210                                             : REM INSERT SBL SET #
7130 GOSUB ' 215                                             : REM INSERT SBL IN TXT
7135 GOSUB ' 110  (D1,D2,D3,D4$,D5)                          : REM SAVE PAGE
7140 GOSUB ' 205  (1)                                        : REM LOAD NXT FORMAT
7145              D3 = 32                                    : REM CLR PAGE INDX
7150              Q4 = 00                                    : REM CLR ADRS INDX
7155 GOSUB ' 210                                             : REM INSERT SBL SET #
7160 GOSUB ' 216                                             : REM CREATE EXERSICES
7165 GOSUB ' 110  (D1,D2,D3,D4$,D5)                          : REM SAVE PAGE
7170 GOSUB ' 205  (1)                                        : REM LOAD NXT FORMAT
7175              D3 = 32                                    : REM CLR PAGE INDX
7180              Q4 = 00                                    : REM CLR ADRS INDX
7185 GOSUB ' 210                                             : REM INSERT SBL SET #
7190 GOSUB ' 217                                             : REM CREATE ANSWERS
7195 GOSUB ' 110  (D1,D2,D3,D4$,D5)                          : REM SAVE PAGE
7200 IF           STR(U2$(U2),16,1)= HEX(00) THEN 7220       : REM END OF LOCAL SBLS
7205 RESTORE      1                                          : REM SET TO BEG FORMAT
7210 GOSUB ' 205  (3)                                        : REM SKIP FIRST FORMAT
7215 GOTO         7110                                       : REM RESTRT SEQUENCE
7220 IF           U1$(U1+1) = HEX(00)       THEN 7245        : REM END OF SIM SBLS
7225              U1 = U1 + 1                                : REM INCR SIM SBL INDX
7230 RESTORE      1                                          : REM SET TO BEG FORMAT
7235 GOSUB ' 205  (1)                                        : REM LOAD FIRST FORMAT
7240 GOTO         7070                                       : REM RESTRT SEQUENCE
7245 GOSUB ' 009                                             : REM RETURN TO AUTO MODE
```

```
7250 REM **
7255 REM **
7260 REM **  Subroutine  ********************************************** 2 1 1
7265 REM **  Abstract >
7270 REM **    To load Similar symbols into a temp storage
7275 REM **    buffer.
7280 REM **  Variables >
7285 REM **    U2$() > sim sbl buffer    ! T1$ > sbl set #
7290 REM **    T2$   > symbol #          ! T3$ > define unpack fmt
7295 REM **    T4$   > search id         ! T2  > sbl set #
7300 REM **    T3    > symbol #          ! U2  > sbl sbl bufr indx
7305 REM **    Q3$() > symbol table      ! Q7$() > symbol id table
7310 REM **    R2$() > srch rturn bufr   ! T1  > storage bufr indx
7315 REM **    R1$() > storage buffer    ! U1  > sim sbl set bufr indx
7320 REM **    Q1$() > sim sbl set bufr !
7325 REM **
7330 REM ***********************************************
7335 DEFFN ' 211                                          : REM ENTRY
7340 DIM          T1$3,T2$4,T3$4,T4$2                     : REM DEFINE WORK BUFFRS
7345 INIT(00)     U2$()                                   : REM INIT SIM TBL
7350              T3$ = HEX(A003A004)                     : REM DEFINE UNPACK FMT
7355              U2  = 1                                 : REM INIT SIM TBL INDX
7360              T1  = VAL(U1$(U1))                      : REM SIM SBL SET INDX
7365              Q0$(1) = "1"                            : REM SKP TBL LOKUP<FLG>
7370 DATA LOAD    BA T #2, (Q3$(T1,3),R0$)Q9$()           : REM LOAD SYMBOL
7375              BIN(R0$(1)) = T1                        : REM CONV SBL # TO BIN
7380 MAT COPY     Q9$()<1,15> TO U2$()<1,15>              : REM STOR SYMBOL
7385 MAT COPY     R0$()<1,1>  TO U2$()<16,1>              : REM STOR SYMBOL #
7390 GOSUB ' 105  (Q3$(T1,6),R0,R1)                       : REM LOAD SIM SBLS
7395              T1 = 1                                  : REM INIT STOR BUFR IDX
7400              U2 = 2                                  : REM INIT SIM TBL INDX
7405 $UNPACK      (F=T3$) R1$()<T1,7> TO T1$,T2$          : REM UNPACK SBL REF #
7410 CONVERT      T1$ TO T2                               : REM CONV SYM SET #
7415 CONVERT      T2$ TO T3                               : REM CONV SYMBOL #
7420 IF           T3 = 9999                    THEN 7490  : REM END OF LIST
7425              BIN(STR(T4$,1,1)) = T2                  : REM CONV TO BINARY
7430              BIN(STR(T4$,2,1)) = T3                  : REM CONV TO BINARY
7435 MAT SEARCH   Q7$(),=STR(T4$,1,2) TO R2$() STEP 2     : REM SRCH SBL ID TBL
7440 IF           R2$() <> HEX(0000)           THEN 7450  : REM SBL ID FOUND
7445              R2$(1) = HEX(0001)                      : REM SET TO SBL #1
7450              T2 = INT(FNR(VAL(R2$(1)))/2) + 1        : REM COMPUTE SBL #
7455              BIN(R0$(1)) = T2                        : REM CONV TO BINARY
7460 DATA LOAD    BA T #2, (Q3$(T2,3),R0$)Q9$()           : REM LOAD SYMBOL
7465 MAT COPY     Q9$()<1,15>TO U2$()<(U2-1)*16+1,15>     : REM LOAD SBL IN BUFR
7470 MAT COPY     R0$()<1,1> TO U2$()<U2*16,1>            : REM LOAD SBL ID IN BUF
7475              U2 = U2 + 1                             : REM INCR SIM SBL INDX
7480              T1 = T1 + 7                             : REM INCR STOR BUFR IDX
7485 IF           T1 < R0                      THEN 7405  : REM NOT END OF BUFR
7490              U2 = 1                                  : REM SET TO BEG OF BUFR
7495 RETURN                                               : REM EXIT
```

```
7500 REM **
7505 REM **
7510 REM **  Subroutine  ********************************************** 2 1 2
7515 REM **  Abstract >
7520 REM **    To save text contained in lower portion of
7525 REM **    page.
7530 REM **  Variables >
7535 REM **    BO$() > temp storage bufr ! Q2$() > page buffer
7540 REM **    RO    > pag bufr lin indx !
7545 REM **
7550 REM ***********************************************
7555 DEFFN ' 212                                          : REM ENTRY
7560 DIM          BO$(8)64                                : REM DEFINE STORAGE
7565 MAT COPY     Q2$()<513,512> TO BO$()<1,512>          : REM STOR LINE
7570 $TRAN        (Q2$()<513,512>,  R4$()<177,4>)00 R     : REM CLR PAG BUFR
7575 GOSUB ' 101  (9,1,8,0)                               : REM CLR CRT SCREEN
7580 RETURN                                               : REM EXIT
```

```
7585 REM **
7590 REM **
7595 REM **  Subroutine  ********************************************** 2 1 3
7600 REM **  Abstract >
7605 REM **    To insert symbol list into page buffer.
7610 REM **  Variables >
7615 REM **    T1 > sim sbls bufr indx  ! T2 > temp text bufr indx
7620 REM **    Q1 > bufr line indx      ! U2$() > sim sbls bufr
7625 REM **    R9$() > temp text bufr   ! Q2$() > page buffer
7630 REM **    T5 > bufr output flag    !
7635 REM **
7640 REM ***********************************************
7645 DEFFN ' 213                                          : REM ENTRY
7650               T1 = 1                                 : REM INIT SIM SBL INDX
7655               T2 = 1                                 : REM INIT TEXT BUF INDX
7660               T5 = 1                                 : REM INIT BUFR FUL FLG
7665               Q1 = 7                                 : REM INIT BUF LINE INDX
7670 INIT(20)      R9$()                                  : REM INIT TEXT BUFR
7675               R9$ = STR(U2$(T1),1,15)                : REM STOR TEMP SYMBOL
7680  IF           T2+LEN(R9$) < 38          THEN 7745    : REM SBL FITS ON LINE
7685               T5 = 0                                 : REM SET END OF LIST
7690  GOSUB ' 100 (Q1,24,40,R9$(1),0)                     : REM LOAD TEMP TEXT BUF
7695               Q1 = Q1 + 1                            : REM INCR BUF LIN INDX
7700  IF           Q1 < 16                   THEN 7730    : REM PAG BUFR NOT FULL
7705  GOSUB '100(16,41,23,"To Continue Co to :nxt0",0)    : REM END OF PAGE MSG
7710  GOSUB ' 110 (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7715  GOSUB ' 101 (2,1,15,0)                              : REM CLR CRT SCREEN
7720  $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R      : REM CLR PAG BUFR
7725               Q1 = 2                                 : REM RESET LINE INDX
7730               T2 = 1                                 : REM RESET TEXT INDX
7735  INIT(20)     R9$()                                  : REM CLR TEXT BUFR
7740  IF           T5 = 2                    THEN 7785    : REM END OF LIST
7745  MAT COPY     U2$()<(T1-1)*16+1,15> TO R9$()<T2,65-T2>  : REM LOAD LIN BUF
7750               T1 = T1 + 1                            : REM INCR SIM SBL INDX
7755               T2 = T2 + LEN(R9$) + 5                 : REM INCR LIN BUF INDX
7760               T5 = 1                                 : REM SET LIN NOT EMPTY
7765 IF            STR(U2$(T1),1,1) <> HEX(00)THEN 7675   : REM MORE SIM SBLS
7770 IF            T5 <> 1                    THEN 7785   : REM BUFR EMPTY
7775               T5 = 2                                 : REM SET END OF LIST
7780 GOTO          7690                                   : REM OUTPUT THE BUFFER
7785 RETURN                                               : REM EXIT
```

```
7790 REM **
7795 REM **
7800 REM **  Subroutine  ********************************************** 2 1 4
7805 REM **  Abstract >
7810 REM **    To save page containing the list of similar
7815 REM **    symbols.
7820 REM **  Variables >
7825 REM **    B0$() > temp storage bufr ! Q2$() > page buffer
7830 REM **    Q1    > pag bufr lin indx !
7835 REM **
7840 REM ***********************************************
7845 DEFFN ' 214                                          : REM ENTRY
7850 IF          16-Q1 >= 2                  THEN 7880    : REM SPACE AVAIABLE
7855 GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):   REM END OF PAG MSG
7860 GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7865 GOSUB ' 101  (2,1,15,0)                              : REM CLR CRT SCREEN
7870 $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R       : REM CLR PAG BUFFER
7875              Q1 = 2                                  : REM RESET PAG LIN INDX
7880 GOSUB ' 100  (Q1  ,1,64,B0$(1),0)                    : REM SAVE LINE
7885 GOSUB ' 100  (Q1+1,1,64,B0$(2),0)                    : REM SAVE LINE
7890              Q1 = Q1 + 3                             : REM INCR LINE INDX
7895 IF          16-Q1 >= 5                  THEN 7925    : REM SPACE AVAIABLE
7900 GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):   REM END OF PAG MSG
7905 GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7910 GOSUB ' 101  (2,1,15,0)                              : REM CLR CRT SCREEN
7915 $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R       : REM CLR PAG BUFFER
7920              Q1 = 2                                  : REM RESET PAG LIN INDX
7925 FOR          T1 = 0 TO 4                             : REM BEG LINE INSERTION
7930 GOSUB ' 100  (Q1+T1,1,64,B0$(T1+3),0)                : REM SAVE LINE
7935 NEXT         T1                                      : REM CONTINUE TO SAVE
7940 GOSUB ' 100  (16  ,1,64,B0$(8),0 )                   : REM SAVE LAST LINE
7945 GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7950 RETURN                                               : REM EXIT
```

```
7585 REM **
7590 REM **
7595 REM **  Subroutine  ********************************************** 2 1 3
7600 REM **  Abstract >
7605 REM **    To insert symbol list into page buffer.
7610 REM **  Variables >
7615 REM **    T1 > sim sbls bufr indx  ! T2 > temp text bufr indx
7620 REM **    Q1 > bufr line indx      ! U2$() > sim sbls bufr
7625 REM **    R9$() > temp text bufr   ! Q2$() > page buffer
7630 REM **    T5 > bufr output flag    !
7635 REM **
7640 REM ***********************************************
7645 DEFFN ' 213                                          : REM ENTRY
7650              T1 = 1                                  : REM INIT SIM SBL INDX
7655              T2 = 1                                  : REM INIT TEXT BUF INDX
7660              T5 = 1                                  : REM INIT BUFR FUL FLG
7665              Q1 = 7                                  : REM INIT BUF LINE INDX
7670 INIT(20)     R9$()                                   : REM INIT TEXT BUFR
7675              R9$ = STR(U2$(T1),1,15)                 : REM STOR TEMP SYMBOL
7680  IF          T2+LEN(R9$) < 38          THEN 7745     : REM SBL FITS ON LINE
7685              T5 = 0                                  : REM SET END OF LIST
7690  GOSUB ' 100 (Q1,24,40,R9$(1),0)                     : REM LOAD TEMP TEXT BUF
7695              Q1 = Q1 + 1                             : REM INCR BUF LIN INDX
7700  IF          Q1 < 16                   THEN 7730     : REM PAG BUFR NOT FULL
7705  GOSUB '100(16,41,23,"To Continue Co to :nxtO",0)    : REM END OF PAGE MSG
7710  GOSUB ' 110 (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7715  GOSUB ' 101 (2,1,15,0)                              : REM CLR CRT SCREEN
7720  $TRAN       (Q2$()<65,960>, R4$()<177,4>)00 R       : REM CLR PAG BUFR
7725              Q1 = 2                                  : REM RESET LINE INDX
7730              T2 = 1                                  : REM RESET TEXT INDX
7735  INIT(20)    R9$()                                   : REM CLR TEXT BUFR
7740  IF          T5 = 2                    THEN 7785     : REM END OF LIST
7745  MAT COPY    U2$()<(T1-1)*16+1,15> TO R9$()<T2,65-T2>  : REM LOAD LIN BUF
7750              T1 = T1 + 1                             : REM INCR SIM SBL INDX
7755              T2 = T2 + LEN(R9$) + 5                  : REM INCR LIN BUF INDX
7760              T5 = 1                                  : REM SET LIN NOT EMPTY
7765 IF           STR(U2$(T1),1,1) <> HEX(00)THEN 7675    : REM MORE SIM SBLS
7770 IF           T5 <> 1                    THEN 7785    : REM BUFR EMPTY
7775              T5 = 2                                  : REM SET END OF LIST
7780 GOTO         7690                                    : REM OUTPUT THE BUFFER
7785 RETURN                                               : REM EXIT
```

```
7790 REM **
7795 REM **
7800 REM **  Subroutine  ********************************************** 2 1 4
7805 REM **  Abstract >
7810 REM **    To save page containing the list of similar
7815 REM **    symbols.
7820 REM **  Variables >
7825 REM **    B0$() > temp storage bufr ! Q2$() > page buffer
7830 REM **    Q1    > pag bufr lin indx !
7835 REM **
7840 REM ********************************************
7845 DEFFN ' 214                                          : REM ENTRY
7850 IF          16-Q1 >= 2                  THEN 7880    : REM SPACE AVAIABLE
7855 GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):   REM END OF PAG MSG
7860 GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7865 GOSUB ' 101  (2,1,15,0)                              : REM CLR CRT SCREEN
7870 $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R       : REM CLR PAG BUFFER
7875              Q1 = 2                                  : REM RESET PAG LIN INDX
7880 GOSUB ' 100  (Q1  ,1,64,B0$(1),0)                    : REM SAVE LINE
7885 GOSUB ' 100  (Q1+1,1,64,B0$(2),0)                    : REM SAVE LINE
7890              Q1 = Q1 + 3                             : REM INCR LINE INDX
7895 IF           16-Q1 >= 5                  THEN 7925    : REM SPACE AVAIABLE
7900 GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):   REM END OF PAG MSG
7905 GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7910 GOSUB ' 101  (2,1,15,0)                              : REM CLR CRT SCREEN
7915 $TRAN        (Q2$()<65,960>, R4$()<177,4>)00 R       : REM CLR PAG BUFFER
7920              Q1 = 2                                  : REM RESET PAG LIN INDX
7925 FOR          T1 = 0 TO 4                             : REM BEG LINE INSERTION
7930 GOSUB ' 100  (Q1+T1,1,64,B0$(T1+3),0)                : REM SAVE LINE
7935 NEXT         T1                                      : REM CONTINUE TO SAVE
7940 GOSUB ' 100  (16  ,1,64,B0$(8),0 )                   : REM SAVE LAST LINE
7945 GOSUB ' 110  (D1,D2,D3,D4$,D5)                       : REM SAVE PAGE
7950 RETURN                                               : REM EXIT
```

```
7955 REM **
7960 REM **
7965 REM **  Subroutine  ********************************************** 2 1 5
7970 REM **  Abstract >
7975 REM **    To insert current similar symbol into page buffer.
7980 REM **  Variables >
7985 REM **    T1 > position to insert at ! R1$() > storage buffer
7990 REM **    Q2$() > page buffer        ! U2$() > sim sbl table
7995 REM **    U2 > sim sbl tbl indx      !
8000 REM **
8005 REM ***********************************************
8010 DEFFN ' 215                                          : REM ENTRY
8015               R9$ = STR(U2$(U2),1,15)                : REM TEMP STOR SYMBOL
8020               T1  = 18 - INT(LEN(R9$)/2)             : REM COMPUTE CENTER
8025 GOSUB ' 100   (11,T1+24,15,R9$,0)                    : REM WRITE SYMBOL
8030 INIT(20)      R1$()                                  : REM CLR STORAGE BUFR
8035 MAT COPY      Q2$()<792,40> TO R1$()<1,40>           : REM MOVE FIXED TEXT
8040 $TRAN         (Q2$()<792,40>, R4$()<177,4>)00 R      : REM CLR TEXT BUFFR
8045 GOSUB ' 101   (13,24,1,0)                            : REM CLR CRT SCREEN
8050 MAT COPY      U2$()<(U2-1)*16+1,15>TO R1$()<14,15>:  REM MOVE IN SYMBOL
8055 GOSUB ' 107   (13,24,60,R0$,0)                       : REM RE-LOAD PAG BUFR
8060 RETURN                                               : REM EXIT
```

```
8065 REM **
8070 REM **
8075 REM **  Subroutine  ********************************************** 2 1 6
8080 REM **  Abstract >
8085 REM **    To create the descrimnation exersices.
8090 REM **  Variables >
8095 REM **    U3$() > position memory bufr  ! U4$() > answers to exersices
8100 REM **    B2$() > random # memory bufr 1! B3$() > random # memory bufr 2
8105 REM **    RO$() > temp srch id          ! R2$() > srch rturn value
8110 REM **    T1 > total # of sim sbls      ! T2 > symbol occurence indx
8115 REM **    T3 > sim sbl tbl loop indx    ! T4 > sbl occurence loop indx
8120 REM **    B1$() > random # mem table    ! B8 > random subscript
8125 REM **    R9$() > temp text buffer      ! Q1 > page bufr line indx
8130 REM **
8135 REM ********************************************
8140 DEFFN ' 216                                        : REM ENTRY
8145 DIM          U3$(28)1,U4$(6)1,B2$(20)1,B3$(20)1    : REM DEFINE WORK BUFRS
8150              RO$(1) = HEX(00)                      : REM DEFINE SRCH ID
8155 MAT SEARCH   U2$(), = RO$(1) TO R2$() STEP 16      : REM SRCH SIM SBL TBL
8160 IF           R2$(1) <> HEX(0000)        THEN 8170  : REM SIM SBL TBL FULL
8165              R2$(1) =  HEX(00A0)                   : REM SET TO BUFR MAX
8170              T1 = INT(FNR(VAL(R2$(1)))/16)         : REM COMPUTE # OF SBLS
8175              T2 = INT(28/T1)                       : REM SET OCCURENCE INDX
8180 IF           T2 <= 6                    THEN 8190  : REM OCCURENCE INDX OK
8185              T2 = 6                                : REM MAX OCCURECNE INDX
8190 INIT(00)     B1$(),B2$(),B3$()                     : REM INIT RND MEM TBLS
8195 INIT(00)     U3$()                                 : REM INIT MEMORY TABLE
8200 FOR          T3 = 1 TO T1                          : REM SET FOR ALL SBLS
8205 FOR          T4 = 1 TO T2                          : REM ACRDING TO OCC IDX
8210 GOSUB ' 200  (28,B1$(1),B8,20)                     : REM GEN RND SUBSCRPT
8215 GOSUB ' 218  (B8,U3$(B8))                          : REM SPREAD SUBSCRPT
8220              U3$(B8) = STR(U2$(T3),16,1)           : REM SET POS MEM BUFR
8225 IF           T3 <> U2                   THEN 8235  : REM NOT PRIMARY SBL
8230              BIN(U4$(T4)) = B8                     : REM STOR ANSWERS
8235 NEXT         T4                                    : REM INCR OCC INDX
8240 NEXT         T3                                    : REM INCR SIM SBL INDX
8245 MAT COPY     B1$() TO B2$()                        : REM SAVE MEM TBL 1
8250 MAT COPY     B2$() TO B1$()                        : REM SET TO MEM TBL 1
8255 GOSUB ' 200  (28,B1$(1),B8,20)                     : REM GEN RND SUBSCRPT
8260 GOSUB ' 218  (B8,U3$(B8))                          : REM SPREAD SUBSCRPT
8265              T3 = B8                               : REM STOR RND SUBSCRPT
8270 MAT COPY     B1$() TO B2$()                        : REM SAVE MEM TBL 1
8275 MAT COPY     B3$() TO B1$()                        : REM SET TO MEM TBL 2
8280 GOSUB ' 200  (QO(2),B1$(1),B8,20)                  : REM GEN RND SUBSCRPT
8285 IF           B8 = VAL(STR(U2$(U2),16,1))THEN 8280: REM RND SBL SAME
8290              BIN(U3$(T3)) = B8                     : REM STOR RND SBL
8295 IF           VAL(B2$(1)) = 28           THEN 8320  : REM POS BUFR FULL
8300 MAT COPY     B1$() TO B3$()                        : REM SAVE MEM TBL 2
8305 IF           VAL(B3$(1)) < QO(2)        THEN 8250  : REM ALL SBLS NOT USED
8310 INIT(00)     B3$()                                 : REM CLR MEMORY BUFR
8315 GOTO         8250                                  : REM CONT TO FILL BUF
8320              QO$(4) = "1"                          : REM SET COL FLAG
8325              Q1 = 2                                : REM INIT PAG LIN INDX
8330              Q2 = 22                               : REM INIT PAG COL INDX
```

```
8335 FOR          T1 = 1 TO 28                                  : REM LOAD THE EXCERSIS
8340 INIT(20)     R9$()                                         : REM CLR TEXT BUFR
8345 CONVERT      T1 TO STR(R9$(1),1,2), (##)                   : REM CONV ITEM #
8350 $TRAN        (R9$()<1,1>, R4$()<185,4>) R                  : REM BLANK FILL NUMBR
8355 MAT COPY     R5$()<8,1> TO R9$()<3,1>                      : REM LOAD DECIMAL
8360              T2 = VAL(U3$(T1))                             : REM FETCH SBL #
8365 DATA LOAD    BA T #2, (Q3$(T2,3),R0$)Q9$()                 : REM LOAD SYMBOL
8370 MAT COPY     Q9$()<1,15> TO R9$()<5,15>                    : REM STOR SYMBOL
8375 IF           Q0$(4) = "1"                THEN 8385         : REM LOAD FRST COL
8380              Q2 = 44                                       : REM SET TO SECOND COL
8385 GOSUB ' 100  (Q1,Q2,18,R9$(1),0)                           : REM STOR IN PAG BUFR
8390              Q1 = Q1 + 1                                   : REM INCR PAG LIN INDX
8395 IF           Q1 <> 16                    THEN 8410         : REM NOT END OF COL
8400              Q1 =  2                                       : REM SET TO TOP OF COL
8405              Q0$(4) = "2"                                  : REM SET COL FLAG
8410 NEXT         T1                                            : REM CONT TO LOAD BUFR
8415 RETURN                                                     : REM EXIT
```

```
8420 REM **
8425 REM **
8430 REM **  Subroutine  ********************************************* 2 1 7
8435 REM **  Abstract >
8440 REM **    To create the answers to the descrimation exerices.
8445 REM **  Variables >
8450 REM **    U5$() > temp sort bufr          ! U6$() > sort work bufr
8455 REM **    U4$() > answer buffer           ! U2$() > sim sbl bufr
8460 REM **    T1 > pag col indx               ! T2 > answer indx
8465 REM **    T3 > value of answer            ! Q2$() > page buffer
8470 REM **
8475 REM *********************************************
8480 DEFFN ' 217                                       : REM ENTRY
8485 DIM          U5$(7)2,U6$(6)2                      : REM DEFINE SORT BUFRS
8490 MAT SORT     U4$()          TO U5$(), U6$()       : REM SORT ANSWER BUFR
8495 INIT(FF)     U5$()                                : REM INIT SORT BUFR
8500 MAT MOVE     U4$(), U6$(1) TO U5$(1)              : REM CREATE SORTED ANS
8505              T1 = 40                              : REM INIT PAG COL INDX
8510              T2 = 1                               : REM INIT ANS BUFR INDX
8515 GOSUB ' 100  (3,24,15,U2$(U2),0)                  : REM INSERT SYMBOL
8520              T3 = VAL(U5$(T2))                    : REM COMPUT VAL OF ANSR
8525 CONVERT      T3 TO R2$(1), (##)                   : REM CONVERT TO DISPLY
8530 $TRAN        (R2$(  )<1,1>, R4$()<185,4>) R       : REM BLANK FILL NUMBR
8535 GOSUB ' 100  (3,T1,2,R2$(1),0)                    : REM WRITE ANSWER
8540              T2 = T2 + 1                          : REM INCR ANS INDX
8545              T1 = T1 + 3                          : REM INCR COL INDX
8550 IF           STR(U5$(T2),1,1) = HEX(FF) THEN 8565 : REM END OF ANS LIST
8555 GOSUB ' 100  (3,T1-1,1,",",0)                     : REM INSERT SEP CHAR
8560 GOTO         8520                                 : REM CONT LOADING ANS
8565              U2 = U2 + 1                          : REM INCR SIM SBL INDX
8570 IF           STR(U2$(U2),16,1)=HEX(00) THEN 8585  : REM END OF SIM SBLS
8575 GOSUB ' 101  (6,1,10,0)                           : REM CLR CRT SCREEN
8580 $TRAN        (Q2$()<321,640>, R4$()<177,4>)00 R   : REM CLR PAG BUFFER
8585 RETURN                                            : REM EXIT
```

```
8590 REM **
8595 REM **
8600 REM **  Subroutine  ********************************************** 2 1 8
8605 REM **  Abstract >
8610 REM **    To mimiumnize the chance that the same two symbols
8615 REM **    will appear in sequence.
8620 REM **  Variables >
8625 REM **    B8 > random subscript      ! U3$() > position memory bufr
8630 REM **
8635 REM ************************************************
8640  DEFFN ' 218 (B8,U3$(B8))                                  : REM ENTRY
8645              B8 = B8 - 7                                   : REM DCRMNT BY CONST
8650  IF          B8 > 0                          THEN 8660     : REM VALID SUBSCRPT
8655              B8 = 28                                       : REM SET TO MAX
8660  IF          U3$(B8) = HEX(00)               THEN 8675     : REM SLOT FOUND
8665              B8 = B8 - 1                                   : REM DCRMNT BY CONST
8670  GOTO        8650                                          : REM BRNCH TO RNG CHK
8675  RETURN                                                    : REM EXIT
```

AUTHOR System: Phase 3 - ATH306.2 - # MAP

7070 - 7240
7110 - 7215
7220 - 7200
7245 - 7220
7405 - 7485
7450 - 7440
7490 - 7420
7675 - 7765
7690 - 7780
7730 - 7700
7745 - 7680
7785 - 7740 7770
7880 - 7850
7925 - 7895
8170 - 8160
8190 - 8180
8235 - 8225
8250 - 8305 8315
8280 - 8285
8320 - 8295
8385 - 8375
8410 - 8395
8520 - 8560
8565 - 8550
8585 - 8570
8650 - 8670
8660 - 8650
8675 - 8660

```
B0$(   - 7560 7565 7880 7885 7930 7940
B1$(   - 8190 8210 8245 8250 8255 8270 8275 8280 8300
B2$(   - 8145 8190 8245 8250 8270 8295
B3$(   - 8145 8190 8275 8300 8305 8310
B8     - 8210 8215 8220 8230 8255 8260 8265 8280 8285 8290 8640 8645 8650 8655
         8660 8665
D1     - 7135 7165 7195 7710 7860 7905 7945
D2     - 7135 7165 7195 7710 7860 7905 7945
D3     - 7080 7115 7135 7145 7165 7175 7195 7710 7860 7905 7945
D4$    - 7135 7165 7195 7710 7860 7905 7945
D5     - 7135 7165 7195 7710 7860 7905 7945
Q0(    - 8280 8305
Q0$(   - 7365 8320 8375 8405
Q1     - 7665 7690 7695 7700 7725 7850 7875 7880 7885 7890 7895 7920 7930 8325
         8385 8390 8395 8400
Q2     - 8330 8380 8385
Q2$(   - 7565 7570 7720 7870 7915 8035 8040 8580
Q3$(   - 7370 7390 7460 8365
Q4     - 7085 7120 7150 7180
Q7$(   - 7435
Q9$(   - 7370 7380 7460 7465 8365 8370
R0     - 7390 7485
R0$    - 7370 7460 8055 8365
R0$(   - 7375 7385 7455 7470 8150 8155
R1     - 7390
R1$(   - 7405 8030 8035 8050
R2$(   - 7435 7440 7445 7450 8155 8160 8165 8170 8525 8530 8535
R4$(   - 7570 7720 7870 7915 8040 8350 8530 8580
R5$(   - 8355
R9$    - 7675 7680 7755 8015 8020 8025
R9$(   - 7670 7690 7735 7745 8340 8345 8350 8355 8370 8385
T1     - 7360 7370 7375 7390 7395 7405 7480 7485 7650 7675 7745 7750 7765 7925
         7930 7935 8020 8025 8170 8175 8200 8335 8345 8360 8410 8505 8535 8545
         8555
T1$    - 7340 7405 7410
T2     - 7410 7425 7450 7455 7460 7655 7680 7730 7745 7755 8175 8180 8185 8205
         8360 8365 8510 8520 8540 8550
T2$    - 7340 7405 7415
T3     - 7415 7420 7430 8200 8220 8225 8240 8265 8290 8520 8525
T3$    - 7340 7350 7405
T4     - 8205 8230 8235
T4$    - 7340 7425 7430 7435
T5     - 7660 7685 7740 7760 7770 7775
U1     - 7050 7220 7225 7360
U1$(   - 7220 7360
U2     - 7200 7355 7400 7465 7470 7475 7490 8015 8050 8225 8285 8515 8565 8570
U2$(   - 7045 7200 7345 7380 7385 7465 7470 7675 7745 7765 8015 8050 8155 8220
         8285 8515 8570
U3$(   - 8145 8195 8215 8220 8260 8290 8360 8640 8660
U4$(   - 8145 8230 8490 8500
U5$(   - 8485 8490 8495 8500 8520 8550
U6$(   - 8485 8490 8500
```

```
AUTHOR System: Phase 3 - ATH306.2 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 9
       - 7245
7070 DEFFN' 50
???? DEFFN' 100
       - 7690 7705 7855 7880 7885 7900 7930 7940 8025 8385 8515 8535 8555
???? DEFFN' 101
       - 7575 7715 7865 7910 8045 8575
???? DEFFN' 102
       - 7065
???? DEFFN' 105
       - 7390
???? DEFFN' 107
       - 8055
???? DEFFN' 110
       - 7135 7165 7195 7710 7860 7905 7945
???? DEFFN' 200
       - 8210 8255 8280
???? DEFFN' 205
       - 7110 7140 7170 7210 7235
???? DEFFN' 210
       - 7090 7125 7155 7185
7335 DEFFN' 211
       - 7075
7555 DEFFN' 212
       - 7095
7645 DEFFN' 213
       - 7100
7845 DEFFN' 214
       - 7105
8010 DEFFN' 215
       - 7130
8140 DEFFN' 216
       - 7160
8480 DEFFN' 217
       - 7190
8640 DEFFN' 218
       - 8215 8260
```

5.19 ATH3Ø6.3

The purpose of the 'ATH3Ø6.3' program is to create the Learn Symbol Definitions (Flowchart and Directions) pages of the learning module.

The remainder of this section presents the 'ATH3Ø6.3' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.3 - CNTRL

```
7000 %## Program > AUTHOR/3  (ATH306.3)   Version > 2.0  2200 T/VP  780322
7010 REM **  Abstract  ************************************************************
7020 REM **                                                                        **
7030 REM **    This program will create the introduction pages to                  **
7040 REM **    the 'Learning The Symbol Definitions'.                              **
7050 REM **                                                                        **
7060 REM ************************************************************************
7070 REM **
7080 REM **
7090  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.3","9000"): REM LOAD DATA FORMATS
7100  DEFFN ' 050                                     : REM DEFINE ENTRY
7110              D3 = 32                             : REM CLR PAGE INDX
7120              Q4 = 00                             : REM CLR ADRS INDX
7130  GOSUB ' 210                                     : REM INSRT SBL SET #
7140  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAV PAG ON DISK
7150  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
7160              D3 = 32                             : REM CLR PAGE INDX
7170              Q4 = 00                             : REM CLR ADRS INDX
7180  GOSUB ' 210                                     : REM INSRT SBL SET #
7190  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
7200  GOSUB ' 010                                     : REM RTURN TO AUTO MODE
```

AUTHOR System: Phase 3 - ATH306.3 - V MAP

```
D1     - 7140 7190
D2     - 7140 7190
D3     - 7110 7140 7160 7190
D4$    - 7140 7190
D5     - 7140 7190
Q4     - 7120 7170
```

```
AUTHOR System: Phase 3 - ATH306.3 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 10
       - 7200
7100 DEFFN' 50
???? DEFFN' 102
       - 7090
???? DEFFN' 110
       - 7140 7190
???? DEFFN' 205
       - 7150
???? DEFFN' 210
       - 7130 7180
```

5.20 ATH3Ø6.4

The purpose of the 'ATH3Ø6.4' program is to create the Learn Symbol Definitions (Definitions and Rest Directions) pages of the learning module.

The remainder of this section presents the 'ATH3Ø6.4' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.4 - CNTRL

```
7000 %## Program > AUTHOR/3  (ATH306.4)   Version > 2.0  2200 T/VP  780322
7010 REM **  Abstract  ************************************************************
7020 REM **                                                                       **
7030 REM **    This program will create the list of Symbols, Definitions          **
7040 REM **    and Memory Aids used in the learning of the symbols.               **
7050 REM **                                                                       **
7060 REM ***************************************************************************
7070 REM **
7080 REM **
7090  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.4","9000"): REM LOAD DATA FORMATS
7100  DEFFN ' 050                                     : REM DEFINE ENTRY
7110            D3 = 32                               : REM CLR PAGE INDX
7120            Q4 = 00                               : REM CLR ADRS INDX
7130            Q1 = 03                               : REM INIT LINE INDX
7140            T1 = U0                               : REM SET SBL TBL INDX
7150  GOSUB ' 210                                     : REM INSRT SBL SET #
7160  GOSUB ' 211                                     : REM SAVE LOWER TEXT
7170  GOSUB ' 212                                     : REM LOAD MEMORY AID
7180  GOSUB ' 213                                     : REM LOAD DEFINITION
7190  GOSUB ' 214                                     : REM WRITE TO PAG BUFR
7200  IF            R0$ <> HEX(0000)        THEN 7170 : REM NOT END OF SBL SET
7210  GOSUB ' 215                                     : REM SAVE PAG ON DISK
7220  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
7230            D3 = 32                               : REM CLR PAGE INDX
7240            Q4 = 00                               : REM CLR ADRS INDX
7250  GOSUB ' 210                                     : REM INSRT SBL SET #
7260  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
7270  GOSUB ' 011                                     : REM RUTRN TO AUTO MODE
```

```
7280 REM **
7290 REM **
7300 REM **  Subroutine  ********************************************** 2 1 1
7310 REM **  Abstract >
7320 REM **    To save last line of text format in a temp storage
7330 REM **    buffer.
7340 REM **  Variables >
7350 REM **    BO$() > temp storage buffer     ! Q2$() > page buffer
7360 REM **    R4$() > conversion buffer       !
7370 REM **
7380 REM ************************************************
7390  DEFFN ' 211                                          : REM ENTRY
7400  DIM          BO$(1)64                                : REM DEFINE WORK BUFR
7410  MAT COPY     Q2$()<961,64> TO BO$()<1,64>            : REM SAVE TEXT
7420  $TRAN        (Q2$()<961,64>, R4$()<177,4>)00 R       : REM CLR TEXT BUFFER
7430  GOSUB ' 101 (16,1,1,0)                               : REM CLR CRT SCREEN
7440  RETURN                                               : REM EXIT
```

```
7450 REM **
7460 REM **
7470 REM **  Subroutine  ********************************************** 2 1 2
7480 REM **  Abstract >
7490 REM **    To load memory aid into temp buffer and determine
7500 REM **    approx. size of text (in lines).
7510 REM **  Variables >
7520 REM **    B2$() > mem aid storage bufr  ! GO$() > register stack
7530 REM **    Q3$() > sbl tbl lokup table   ! R1$() > storage buffer
7540 REM **    RO > buffer size              ! Q1 > output page lin indx
7550 REM **    Q2$() > page buffer           ! R4$() > conversion buffer
7560 REM **    T1 > symbol table indx        ! R1 > item id
7570 REM **
7580 REM ***********************************************
7590  DEFFN ' 212                                          : REM ENTRY
7600  DIM          B2$(16)32                               : REM DEFINE WORK BUFR
7610               GO$(1) = "1"                            : REM SET NULL LOOKUP
7620  GOSUB ' 105 (Q3$(T1,5),RO,R1)                        : REM FETCH MEMORY AID
7630  MAT COPY     R1$() TO B2$()                          : REM LOAD TEMP BUFFR
7640  IF           RO <= 512                  THEN 7660    : REM BUFR SIZ OK
7650               RO  = 512                               : REM SET TO BUFR MAX
7660               RO = INT(RO/18) + 1                     : REM CMPUTE SIZE
7670  IF           16-Q1 >= RO                THEN 7730    : REM SPACE AVAIABLE
7680  GOSUB ' 100 (16,41,23,"To Continue Go to :nxtO",0): REM INSRT EOP MSG
7690  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAG ON DISK
7700  GOSUB ' 101 (3,1,14,0)                               : REM CLR CRT SCREEN
7710  $TRAN        (Q2$()<129,896> , R4$()<177,4>)00 R     : REM CLR TEXT BUFFER
7720               Q1 = 03                                 : REM RESET LINE INDX
7730  RETURN                                               : REM EXIT
```

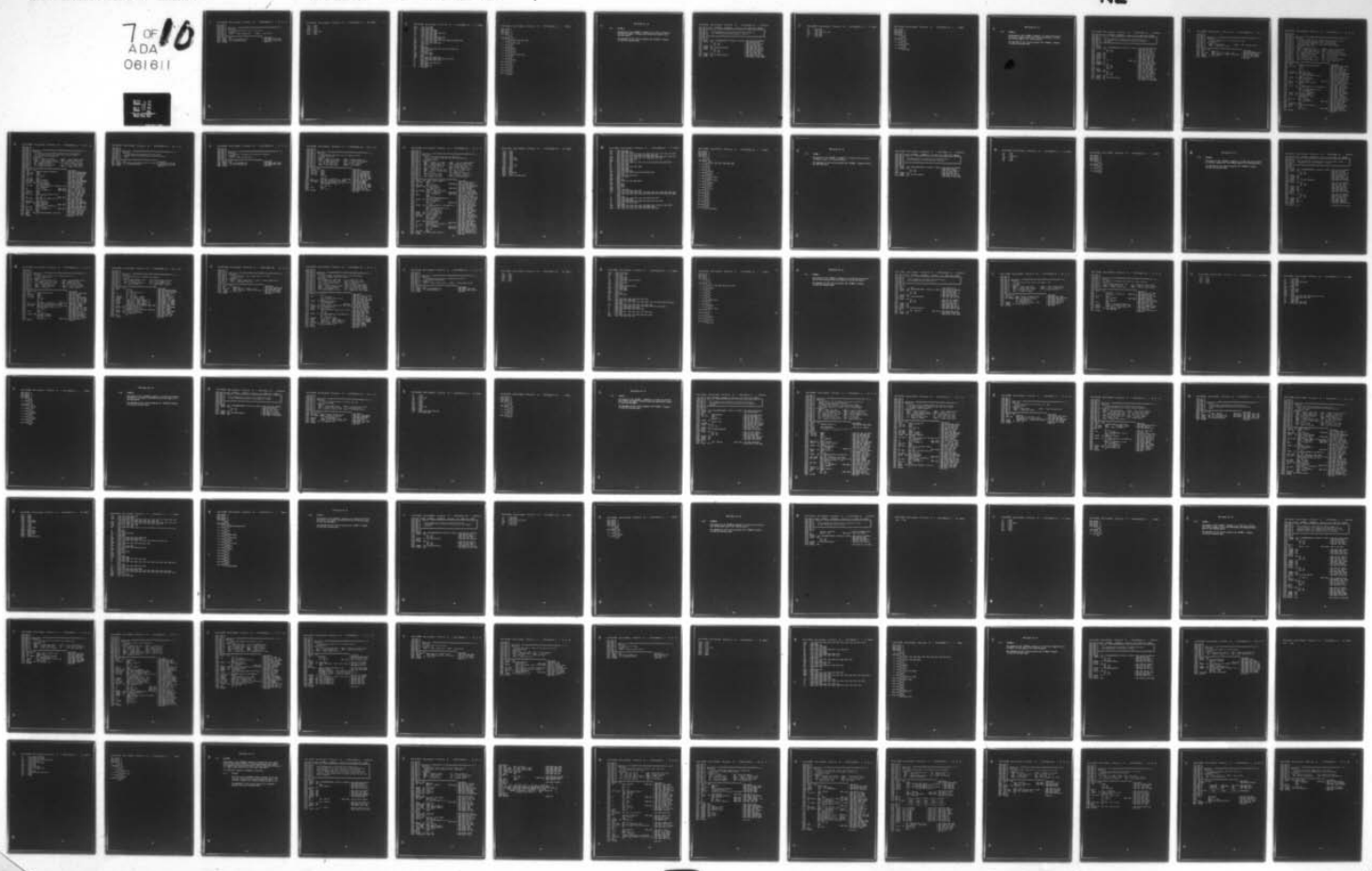
```
7740 REM **
7750 REM **
7760 REM **  Subroutine  ********************************************** 2 1 3
7770 REM **  Abstract >
7780 REM **    To load definition into temp buffer and determine
7790 REM **    approx. size of text (in lines).
7800 REM **  Variables >
7810 REM **    Q0$() > register stack
7820 REM **    Q3$() > sbl tbl lokup table    ! R1$() > storage buffer
7830 REM **    R0 > buffer size               ! Q1 > output page lin indx
7840 REM **    Q2$() > page buffer            ! R4$() > conversion buffer
7850 REM **    T1 > symbol table indx         ! R1 > item id
7860 REM **    T5 > scaled buffer size        !
7870 REM **
7880 REM ***********************************************
7890  DEFFN ' 213                                          : REM ENTRY
7900              Q0$(1) = "1"                             : REM SET NULL LOOKUP
7910  GOSUB ' 105 (Q3$(T1,4),R0,R1)                        : REM FETCH DEFINITION
7920              R0 = INT(R0/18) + 1                      : REM CMPUTE SIZE
7930              T5 = R0                                  : REM SAVE BUFR SIZE
7940  IF          16-Q1 >= R0                THEN 8000 : REM SPACE AVAIABLE
7950  GOSUB ' 100 (16,41,23,"To Continue Go to :nxt0",0): REM INSRT EOP MSG
7960  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAG ON DISK
7970  GOSUB ' 101 (3,1,14,0)                               : REM CLR CRT SCREEN
7980  $TRAN       (Q2$()<129,896> , R4$()<177,4>)00 R : REM CLR TEXT BUFFER
7990              G1 = 03                                  : REM RESET LINE INDX
8000  RETURN                                               : REM EXIT
```

```
8010 REM **
8020 REM **
8030 REM **  Subroutine  ********************************************** 2 1 4
8040 REM **  Abstract >
8050 REM **    To Insert the definition, memory aid, and symbol into
8060 REM **    page buffer.
8070 REM **  Variables >
8080 REM **    M() > line number mem bufr  ! Q2$() > page buffer
8090 REM **    T2  > page bufr lin indx    ! Q1 > page bufr lin indx
8100 REM **    T1  > symbol table indx     ! U0 > symbol table indx
8110 REM **    Q3$() > symbl lookup table  ! Q7$() > symbl id table
8120 REM **    R5$() > single char storage ! Q9$() > i/o buffer
8130 REM **    R9$() > scratch buffer      ! T3 > temp col indx
8140 REM **    T4    > symbol length       ! R0$ > eof flag
8150 REM **    T5    > bufr siz of deffn   ! T6  > left margin adjustmnt
8160 REM **
8170 REM ***********************************************
8180  DEFFN ' 214                                          : REM ENTRY
8190  DIM          M(2)                                    : REM DEFINE WORK BUFR
8200               T2 = Q1                                 : REM STOR LINE INDX
8210               T6 = 6                                  : REM SET LEFT MARGIN
8220  IF           Q0$(9) <> "Y"              THEN 8270    : REM OPTION NOT SEL
8230  IF           T5 <> 1                    THEN 8270    : REM NOT ONE LINE
8240  MAT COPY     R1$()<1,18> TO R9$()<1,64>              : REM STOR ONE LINE
8250               T4 = LEN(R9$(1))                        : REM CMPUT LEN OF DEFN
8260               T6 = T6 + (20-T4)                       : REM CMPUT MARGN ADJUST
8270  GOSUB ' 107 (T2,T6,26,Q3$(T1,4),VAL(Q0$(10)))       : REM INSRT DEFINITION
8280               M(1) = Q1                               : REM STOR LINE INDX
8290  MAT COPY     B2$() TO R1$()                          : REM FETCH MEM AID
8300  GOSUB ' 107 (T2,43,63,Q3$(T1,5),VAL(Q0$(10)))       : REM INSRT MEM AID
8310               M(2) = Q1                               : REM STOR LINE INDX
8320  GOSUB ' 100 (T2,4,1,R5$(9),0)                        : REM INSRT ITEM FLAG
8330  DATA LOAD    BA T #2, (Q3$(T1,3),R0$)Q9$()           : REM FETCH SYMBOL
8340  MAT COPY     Q9$()<1,15> TO R9$()<1,64>              : REM LOAD INTO BUFR
8350               T4 = LEN(R9$(1))                        : REM CMPUTE LENGTH
8360               T3 = 34 - INT(T4/2)                     : REM CMPUTE COL INDX
8370  GOSUB ' 100 (T2,T3,T4,R9$(1),0)                      : REM INSRT SYMBOL
8380               Q1 = M(1) + 1                           : REM SET LINE INDX
8390  IF           M(1) > M(2)                THEN 8410    : REM IF M(1) MAXIUM
8400               Q1 = M(2) + 1                           : REM RESET LINE INDX
8410               T1 = T1 + 1                             : REM INCR SBL INDX
8420  IF           Q7$(U0,1) =  Q7$(T1,1)     THEN 8440    : REM NOT END OF SET
8430               R0$ = HEX(0000)                         : REM SET EOF FLAG
8440  RETURN                                               : REM EXIT
```

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U)
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

7 OF 10
ADA
061611



```
8450 REM **
8460 REM **
8470 REM **  Subroutine  ********************************************** 2 1 5
8480 REM **  Abstract >
8490 REM **    To save final page of text on the disk.
8500 REM **  Variables >
8510 REM **    BO$() > temp storage buffr    ! Q2$() > page buffer
8520 REM **
8530 REM ********************************************
8540  DEFFN ' 215                                          : REM ENTRY
8550  GOSUB ' 100 (16,1,64,BO$(1),0)                       : REM INSRT LAST LINE
8560  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAG ON DISK
8570  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH306.4 - # MAP

7170 - 7200
7660 - 7640
7730 - 7670
8000 - 7940
8270 - 8220 8230
8410 - 8390
8440 - 8420

```
B0$(   - 7400 7410 8550
B2$(   - 7600 7630 8290
D1     - 7260 7690 7960 8560
D2     - 7260 7690 7960 8560
D3     - 7110 7230 7260 7690 7960 8560
D4$    - 7260 7690 7960 8560
D5     - 7260 7690 7960 8560
M(     - 8190 8280 8310 8380 8390 8400
Q0$(   - 7610 7900 8220 8270 8300
Q1     - 7130 7670 7720 7940 7990 8200 8280 8310 8380 8400
Q2$(   - 7410 7420 7710 7980
Q3$(   - 7620 7910 8270 8300 8330
Q4     - 7120 7240
Q7$(   - 8420
Q9$(   - 8330 8340
R0     - 7620 7640 7650 7660 7670 7910 7920 7930 7940
R0$    - 7200 8330 8430
R1     - 7620 7910
R1$(   - 7630 8240 8290
R4$(   - 7420 7710 7980
R5$(   - 8320
R9$(   - 8240 8250 8340 8350 8370
T1     - 7140 7620 7910 8270 8300 8330 8410 8420
T2     - 8200 8270 8300 8320 8370
T3     - 8360 8370
T4     - 8250 8260 8350 8360 8370
T5     - 7930 8230
T6     - 8210 8260 8270
U0     - 7140 8420
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 11
       - 7270
7100 DEFFN' 50
???? DEFFN' 100
       - 7680 7950 8320 8370 8550
???? DEFFN' 101
       - 7430 7700 7970
???? DEFFN' 102
       - 7090
???? DEFFN' 105
       - 7620 7910
???? DEFFN' 107
       - 8270 8300
???? DEFFN' 110
       - 7260 7690 7960 8560
???? DEFFN' 205
       - 7220
???? DEFFN' 210
       - 7150 7250
7390 DEFFN' 211
       - 7160
7590 DEFFN' 212
       - 7170
7890 DEFFN' 213
       - 7180
8180 DEFFN' 214
       - 7190
8540 DEFFN' 215
       - 7210
```

5.21 ATH3Ø6.5

The purpose of the 'ATH3Ø6.5' program is to create the Practice Defining the Symbols (Flowchart and Directions) pages of the learning module.

The remainder of this section presents the 'ATH3Ø6.5' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.5 - CNTRL

```
7000 %## Program > AUTHOR/3   (ATH306.5)   Version > 2.0  2200 T/VP  780322
7010 REM **  Abstract  ************************************************************
7020 REM **                                                                        **
7030 REM **    This program will create the introduction frames to                 **
7040 REM **    the 'Practice Defining the Symbols' section.                        **
7050 REM **                                                                        **
7060 REM ************************************************************************
7070 REM **
7080 REM **
7090  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.5","9000"): REM LOAD DATA FORMATS
7100  DEFFN ' 050                                     : REM DEFINE ENTRY
7110              D3 = 32                             : REM CLR PAG INDX FLG
7120              Q4 = 00                             : REM CLR ADRS INDX
7130  GOSUB ' 210                                     : REM INSRT SBL SET #
7140  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
7150  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
7160              D3 = 32                             : REM CLR PAG INDX FLG
7170              Q4 = 00                             : REM CLR ADRS INDX
7180  GOSUB ' 210                                     : REM INSRT SBL SET #
7190  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
7200  GOSUB ' 012                                     : REM RTURN TO AUTO MODE
```

AUTHOR System: Phase 3 - ATH306.5 - V MAP

```
D1     - 7140 7190
D2     - 7140 7190
D3     - 7110 7140 7160 7190
D4$    - 7140 7190
D5     - 7140 7190
Q4     - 7120 7170
```

```
AUTHOR System: Phase 3 - ATH306.5 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 12
       - 7200
7100 DEFFN' 50
???? DEFFN' 102
       - 7090
???? DEFFN' 110
       - 7140 7190
???? DEFFN' 205
       - 7150
???? DEFFN' 210
       - 7130 7180
```

5.22 ATH3Ø6.6

The purpose of the 'ATH3Ø6.6' program is to create the Practice Defining the Symbols (Exercises, Practice Directions, and Rest Directions) pages of the learning module.

The remainder of this section presents the 'ATH3Ø6.6' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.6 - CNTRL

```
7000 %## Program > AUTHOR/3  (ATH306.6)   Version > 2.0  2200 T/VP  780322
7005 REM **  Abstract  ************************************************************
7010 REM **                                                                       **
7015 REM **    This program will construct the practice excersies                 **
7020 REM **    and answers for the 'Practice Defining the Symbols'                **
7025 REM **    section.                                                           **
7030 REM ************************************************************************
7035 REM **
7040 REM **
7045  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.6","9000"): REM LOAD DATA FORMATS
7050  DEFFN ' 050                                     : REM DEFINE ENTRY
7055              D3 = 00                             : REM SET PAG INDX FLG
7060              Q4 = 00                             : REM CLR ADRS INDX
7065  GOSUB ' 216                                     : REM GEN SBL SET TBL
7070  GOSUB ' 210                                     : REM INSRT SBL SET #
7075  GOSUB ' 211                                     : REM SAVE LOWER TEXT
7080  GOSUB ' 212                                     : REM INSRT SBL LIST
7085  GOSUB ' 213                                     : REM INSRT SBL TEST
7090  IF            T9 = 0                THEN 7125   : REM IF END OF SEQU
7095  GOSUB ' 214                                     : REM SAVE PAGE ON DSK
7100  RESTORE       1                                 : REM RSTR FORMT INDX
7105  GOSUB ' 205 (1)                                 : REM RE-LOAD FORMAT
7110              D3 = 32                             : REM CLR PAG INDX FLG
7115              Q4 = 00                             : REM CLR ADRS INDX
7120  GOTO          7070                              : REM CREATE ANOTHR FMT
7125  GOSUB ' 215                                     : REM SAVE LAST PAGE
7130  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
7135              D3 = 32                             : REM CLR PAG INDX FLG
7140              Q4 = 00                             : REM CLR ADRS INDX
7145  GOSUB ' 210                                     : REM INSRT SBL SET #
7150  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
7155  GOSUB ' 013                                     : REM RTURN TO AUTO MODE
```

```
7160 REM **
7165 REM **
7170 REM **  Subroutine  ********************************************** 2 1 1
7175 REM **  Abstract >
7180 REM **    To save last line of text buffer in a temp
7185 REM **    storage buffer.
7190 REM **  Variables >
7195 REM **    Q2$() > page buffer          ! BO$() > temp storage buffer
7200 REM **    R4$() > conversion buffer    !
7205 REM **
7210 REM ***********************************************
7215  DEFFN ' 211                                                  : REM ENTRY
7220  DIM          BO$(1)64                                        : REM DEFINE WORK BUFRS
7225  MAT COPY     Q2$()<961,64> TO BO$()<1,64>                    : REM LOAD WRK BUFR
7230  $TRAN        (Q2$()<961,64> , R4$()<177,4>)00 R              : REM CLR TEXT BUFFER
7235  GOSUB ' 101 (16,1,1,0)                                       : REM CLR CRT SCREEN
7240  RETURN                                                       : REM EXIT
```

```
7245 REM **
7250 REM **
7255 REM **  Subroutine  ********************************************** 2 1 2
7260 REM **  Abstract >
7265 REM **    To insert a selection of symbols and symbol definitions
7270 REM **    from the symbol set table into the page buffer.
7275 REM **    If insufficient symbols to fill space or symbol
7280 REM **    definition too long fill space with others symbols
7285 REM **    from same symbol set.
7290 REM **  Variables >
7295 REM **    U2$() > symbl set table      ! U3$() > sbl set lokup table
7300 REM **    B1$() > systm memory buffr   ! B2$() > memory buffr #1
7305 REM **    U3 > sbl set lokup tbl indx  ! T9 > end of sequence flag
7310 REM **    Q1 > page buffer line indx   ! T7 > total sbls in curnt set
7315 REM **    T1 > symbol table index      ! Q0$() > stack register flags
7320 REM **    R0 > storage bufr size       ! R1 > item id indx
7325 REM **    T3 > bufr line indx (temp)   ! R9$() > scratch bufr
7330 REM **    T5 > symbol length           ! T6 > bufr col indx (center)
7335 REM **    R5$() > single char buffer   ! B8 > random subscript
7340 REM **
7345 REM ************************************************
7350  DEFFN ' 212                                          : REM ENTRY
7355  DIM          U3$(8)1,U2$(15)16,B2$(5)1               : REM DEFINE WORK BUFRS
7360  INIT(00)     U3$()                                   : REM INIT BUFFR
7365               U3 = 1                                  : REM INIT LOKUP INDX
7370               T9 = 1                                  : REM INIT EOF FLAG
7375               Q1 = 9                                  : REM INIT LINE INDX
7380  MAT COPY     B2$() TO B1$()                          : REM RESET SYS MEM BUFR
7385  GOSUB ' 200 (T7,B1$(1),B8,5)                         : REM GEN RND SUBSCRPT
7390               T1 = VAL(STR(U2$(B8),16,1))             : REM CMPUT SBL TBL INDX
7395               Q0$(1) = "1"                            : REM SET NUL LOKUP FLG
7400  GOSUB ' 105 (Q3$(11,4),R0,R1)                        : REM FETCH DEFINITION
7405               R0 = INT(R0/36) + 1                     : REM CMPUT APPROX SIZ
7410  IF           16-Q1 >= R0                THEN 7425    : REM IF ENOUGH SPACE
7415  GOSUB ' 217 (U2$(1),U3$(1),B2$(1))                   : REM FILL REMAING SPACE
7420  GOTO         7510                                    : REM BRANCH TO EXIT
7425               T3 = Q1                                 : REM STORE LINE INDX
7430  GOSUB ' 107 (T3,24,62,Q3$(11,4),VAL(Q0$(10)))        : REM INSRT DEFINITION
7435               Q1 = Q1 + 1                             : REM INCR LINE INDX
7440               R9$(1) = STR(U2$(B8),1,15)              : REM STORE SYMBOL
7445               T5 = LEN(R9$(1))                        : REM CMPUT SBL LENGTH
7450               T6 = 13 - INT(T5/2)                     : REM CMPUT COL CENTER
7455  GOSUB ' 100 (T3,16,T5,R9$(1),0)                      : REM INSRT SYMBOL
7460  GOSUB ' 100 (T3,22,1,R5$(9),0)                       : REM INSRT ITEM FLAG
7465               BIN(U3$(U3)) = B8                       : REM STOR SBL SBSCRPT
7470               U3 = U3 + 1                             : REM INCR LOKUP TBL IDX
7475  MAT COPY     B1$() TO B2$()                          : REM UPDAT MEM TBL #1
7480  IF           T7 <> VAL(B2$(1))          THEN 7505    : REM IF END OF SEQU
7485               T9 = 0                                  : REM SET EOF FLAG
7490  INIT(00)     B2$()                                   : REM CLR MEM TBL #1
7495  GOSUB ' 217 (U2$(1),U3$(1),B2$(1))                   : REM FIL REMAING SPACE
7500  GOTO         7510                                    : REM BRANCH TO EXIT
7505  IF           Q1 < 16                    THEN 7385    : REM FETCH NEW SBL
7510               U3 = U3 - 1                             : REM CMPUT TOTAL SBLS
7515  RETURN                                               : REM EXIT
```

```
7520 REM **
7525 REM **
7530 REM **  Subroutine  ********************************************** 2 1 3
7535 REM **  Abstract >
7540 REM **    To create a list of symbols selected from those symbols
7545 REM **    used in lower portion of frame. The list is 6 lines
7550 REM **    of randomly ordered symbols.
7555 REM **  Variables >
7560 REM **    B4$() > temp line buffer       ! B5$() > curnt symbol buffer
7565 REM **    B1$() > system memory buffer   ! B2$() > memory buffer #1
7570 REM **    Q0$() > stack register buffer  ! U3 > total symbls in list
7575 REM **    B8 > random subscript          ! T1 > bufr line indx (loop)
7580 REM **    T3 > temp line bufr index      ! T4 > sbl list lokup index
7585 REM **    T5 > length of symbol          ! R5$() > single char buffer
7590 REM **
7595 REM ***********************************************
7600  DEFFN ' 213                                          : REM ENTRY
7605  DIM          B4$(1)64,B5$(1)15,B3$(5)1               : REM DEFINE WORK BUFRS
7610  INIT(00)     B1$()                                   : REM INIT SYS MEM BUFR
7615  INIT(00)     B3$()                                   : REM INIT MEM BUFR #1
7620  FOR          T1 = 2 TO 6                             : REM BEGIN LINE INDX
7625               T3 = 24                                 : REM INIT COL INDX
7630  INIT(20)     B4$()                                   : REM CLR LINE BUFR
7635  MAT COPY     B3$() TO B1$()                          : REM RESET SYS MEM BUFR
7640  MAT COPY     B1$() TO B3$()                          : REM UPDAT MEM BUFR #1
7645  GOSUB ' 200 (U3,B1$(1),B8,5)                         : REM GEN RND SBSCRPT
7650               T4 = VAL(U3$(B8))                       : REM FETCH TBL INDX
7655  MAT COPY     U2$()<(T4-1)*16+1,15> TO B5$()<1,15>:   REM FETCH SYMBOL
7660               T5 = LEN(B5$(1))                        : REM CMPUT SBL LENGTH
7665  IF           T3+T5 <= 60                THEN 7685    : REM IF LINE NOT FULL
7670  IF           U3 <> VAL(B1$(1))          THEN 7645    : REM IF LIST NOT EMPTY
7675  INIT(00)     B1$()                                   : REM CLR SYS MEM BUFR
7680  GOTO         7750                                    : REM WRITE CURNT LINE
7685  MAT COPY     B5$()<1,T5> TO B4$()<T3,T5>             : REM LOAD LINE BUFR
7690               T3 = T3 + T5                            : REM INCR LINE INDX
7695  IF           T3 > 56                    THEN 7750    : REM IF LINE FULL
7700  MAT COPY     R5$()<9,1> TO B4$()<T3+1,2>             : REM LOAD SEP CHAR
7705               T3 = T3 + 3                             : REM INCR LINE INDX
7710  MAT COPY     B3$() TO B1$()                          : REM RESET SYS MEM BUFR
7715  GOSUB ' 202 (B1$,B1$(1),B8)                          : REM UPDAT SYS MEM BUFR
7720  IF           B1$ <> HEX(00)             THEN 7735    : REM DUPLICIAT SBL
7725               BIN(B1$(1)) = VAL(B1$(1)) + 1           : REM INCR BITS ON COUNT
7730  MAT COPY     B1$() TO B3$()                          : REM UPDAT MEM BUFR #1
7735  IF           U3 <> VAL(B3$(1))          THEN 7640    : REM IF NOT END OF LIST
7740  INIT(00)     B1$()                                   : REM CLR SYS MEM BUFR
7745  GOTO         7640                                    : REM START LIST OVER
7750  GOSUB ' 100 (T1,24,40,STR(B4$(1),24,40),0)           : REM WRITE LINE BUFR
7755  NEXT         T1                                      : REM INCR LINE INDX
7760  RETURN                                               : REM EXIT
```

```
7765 REM **
7770 REM **
7775 REM **  Subroutine  ********************************************** 2 1 4
7780 REM **  Abstract >
7785 REM **    To save page current page on disk and insert
7790 REM **    message indicating that section continues on next
7795 REM **    page.
7800 REM **
7805 REM ********************************************
7810  DEFFN ' 214                                          : REM ENTRY
7815  GOSUB ' 100 (16,41,23,"To Continue Go to :nxtO",0): REM END OF PAG MSG
7820  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAG ON DSK
7825  RETURN                                               : REM EXIT
```

```
7830 REM **
7835 REM **
7840 REM **  Subroutine  ********************************************** 2 1 5
7845 REM **  Abstract >
7850 REM **    To save the last page of the practice sequence.
7855 REM **  Variables >
7860 REM **    BO$() > temp storage buffer    !
7865 REM **
7870 REM ***********************************************
7875  DEFFN ' 215                                        : REM ENTRY
7880  GOSUB ' 100 (16,1,64,BO$(1),0)                     : REM INSRT LAST LINE
7885  GOSUB ' 110 (D1,D2,D3,D4$,D5)                      : REM SAVE PAG ON DSK
7890  RETURN                                             : REM EXIT
```

```
7895 REM **
7900 REM **
7905 REM **  Subroutine  ********************************************** 2 1 6
7910 REM **  Abstract >
7915 REM **    To generate and store the list of symbols in the
7920 REM **    current symbol set.
7925 REM **  Variables >
7930 REM **    U2$() > symbl list table     ! B2$() > memory buffer #1
7935 REM **    B3$() > memory buffer #2     ! B1$() > system memory buffer
7940 REM **    T1 > sbl tbl lokup indx      ! U0 > sbl tbl lokup indx
7945 REM **    T7 > sbl list tbl indx       ! Q7$() > sbl id table
7950 REM **    Q3$() > sbl lokup table      ! Q9$() > i/o buffer
7955 REM **
7960 REM ***********************************************
7965  DEFFN ' 216                                          : REM ENTRY
7970  INIT(00)    U2$()                                    : REM INIT TABLE BUFR
7975  INIT(00)    B1$()                                    : REM CLR SYS MEM BUFR
7980  INIT(00)    B2$()                                    : REM CLR MEM BUFR #1
7985  INIT(00)    B3$()                                    : REM CLR MEM BUFR #2
7990              T1 = U0                                  : REM INIT SBL TBL INDX
7995              T7 = 1                                   : REM INIT SBL LST INDX
8000  IF          Q7$(U0,1) <> Q7$(T1,1)    THEN 8035      : REM END OF SBL SET
8005  DATA LOAD   BA T #2, (Q3$(T1,3),R0$) Q9$()           : REM FETCH SYMBOL
8010  MAT COPY    Q9$()<1,15> TO U2$()<(T7-1)*16+1,15>     : REM STOR SYMBOL
8015              BIN(STR(U2$(T7),16,1)) = T1              : REM STOR SBL TBL INDX
8020              T1 = T1 + 1                              : REM INCR SBL TBL INDX
8025              T7 = T7 + 1                              : REM INCR SBL LST INDX
8030  GOTO        8000                                     : REM FETCH NXT SBL
8035              T7 = T7 - 1                              : REM CMPUT TOTAL SBLS
8040  RETURN                                               : REM EXIT
```

```
8045 REM **
8050 REM **
8055 REM **  Subroutine  ********************************************** 2 1 7
8060 REM **  Abstract >
8065 REM **    To fill the space remaining on a page with a
8070 REM **    symbol from current set that has not been inserted
8075 REM **    on current page.
8080 REM **  Variables >
8085 REM **    U2$() > symbol list table    ! U3$() > sbl set lookup table
8090 REM **    B2$() > memory buffer #1     ! B3$() > memory buffer #2
8095 REM **    B1$() > system memory bufr   ! Q1 > output page line indx
8100 REM **    T7 > total sbls in list      ! B8 > random subscript
8105 REM **    R0$() > search id            ! R2$() > srch return buffer
8110 REM **    Q0$() > stack register bufr  ! T1 > sbl tbl lokup indx
8115 REM **    Q3$() > sbl tbl lokup table  ! R0 > buffer size
8120 REM **    R1 > item id                 ! T3 > temp bufr line indx
8125 REM **    R9$() > scratch buffer       ! T5 > length of symbol
8130 REM **    T6 > buffer col index        ! R5$() > single char bufr
8135 REM **    B1$ > memory indx flg        ! T9 > end of list flag
8140 REM **
8145 REM ***********************************************
8150  DEFFN ' 217 (U2$(1),U3$(1),B2$(1))                  : REM ENTRY
8155  INIT(00)    B3$()                                   : REM CLR MEM BUFR #2
8160  IF          Q1 >= 16                  THEN 8335     : REM IF PAGE FULL
8165  MAT COPY    B3$() TO B1$()                          : REM RESET SYS MEM BUFR
8170  GOSUB ' 200 (T7,B1$(1),B8,5)                        : REM GEN RND SBSCRPT
8175  IF          T7 < VAL(B1$(1))          THEN 8335     : REM END OF SBLS
8180              BIN(R0$(1)) = B8                        : REM CONV TO SRCH ID
8185  MAT SEARCH  U3$(), = R0$(1) TO R2$()                : REM SRCH FOR SAM SBL
8190  IF          R2$(1) = HEX(0000)        THEN 8205     : REM IF NO DUPLICIATES
8195  IF          T7 <> VAL(B1$(1))         THEN 8170     : REM IF ALL SBLS ON PAG
8200  GOTO        8335                                    : REM BRANCH TO EXIT
8205              Q0$(1) = "1"                            : REM SET NULL LOKUP FLG
8210              T1 = VAL(STR(U2$(B8),16,1))             : REM FETCH LST TBL INDX
8215  GOSUB ' 105 (Q3$(T1,4),R0,R1)                       : REM FETCH DEFINITION
8220              R0 = INT(R0/36) + 1                     : REM CMPUT APPROX SIZ
8225  IF          16-Q1 < R0                THEN 8170     : REM SBL DOES NOT FIT
8230              T3 = Q1                                 : REM STORE LINE INDX
8235  GOSUB ' 107 (T3,24,62,Q3$(T1,4),VAL(Q0$(10)))       : REM INSRT DEFINITION
8240              Q1 = Q1 + 1                             : REM INCR LINE INDX
8245              R9$(1) = STR(U2$(B8),1,15)              : REM STORE SYMBOL
8250              T5 = LEN(R9$(1))                        : REM CMPUT SBL LENGTH
8255              T6 = 13 - INT(T5/2)                     : REM CMPUT COL CENTER
8260  GOSUB ' 100 (T3,T6,T5,R9$(1),0)                     : REM INSRT SYMBOL
8265  GOSUB ' 100 (T3,22,1,R5$(9),0)                      : REM INSRT ITEM FLAG
8270  MAT COPY    B1$() TO B3$()                          : REM UPDAT MEM BUFR #2
8275  MAT COPY    B2$() TO B1$()                          : REM RESET SYS MEM BUFR
8280              BIN(U3$(U3)) = B8                       : REM ADD TO INDX TBL
8285              U3 = U3 + 1                             : REM INCR TBL INDX
8290  GOSUB ' 202 (B1$,B1$(1),B8)                         : REM ADD TO MEM BUFFR
8295  IF          B1$ <> HEX(00)            THEN 8165     : REM DUPLICATE SBL
8300              BIN(B1$(1)) = VAL(B1$(1)) + 1           : REM INCR SYS COUNTER
8305  MAT COPY    B1$() TO B2$()                          : REM UPDAT MEM BUFR #1
8310  IF          T7 <> VAL(B2$(1))         THEN 8160     : REM MORE SBLS TO SCAN
8315  IF          T9 =  0                   THEN 8335     : REM END OF SBL LIST
8320              T9 = 0                                  : REM SET END OF PAG FLG
8325  INIT(00)    B2$()                                   : REM CLR MEM BUFR #1
8330  GOSUB ' 217 (U2$(1),U3$(1),B2$(1))                  : REM FILL REMAING SPACE
8335  RETURN                                              : REM EXIT
```

AUTHOR System: Phase 3 - ATH306.6 - # MAP

```
7070  - 7120
7125  - 7090
7385  - 7505
7425  - 7410
7505  - 7480
7510  - 7420 7500
7640  - 7735 7745
7645  - 7670
7685  - 7665
7735  - 7720
7750  - 7680 7695
8000  - 8030
8035  - 8000
8160  - 8310
8165  - 8295
8170  - 8195 8225
8205  - 8190
8335  - 8160 8175 8200 8315
```

```
B0$(   - 7220 7225 7880
B1$    - 7715 7720 8290 8295
B1$(   - 7380 7385 7475 7610 7635 7640 7645 7670 7675 7710 7715 7725 7730 7740
         7975 8165 8170 8175 8195 8270 8275 8290 8300 8305
B2$(   - 7355 7380 7415 7475 7480 7490 7495 7980 8150 8275 8305 8310 8325 8330
B3$(   - 7605 7615 7635 7640 7710 7730 7735 7985 8155 8165 8270
B4$(   - 7605 7630 7685 7700 7750
B5$(   - 7605 7655 7660 7685
B8     - 7385 7390 7440 7465 7645 7650 7715 8170 8180 8210 8245 8280 8290
D1     - 7150 7820 7885
D2     - 7150 7820 7885
D3     - 7055 7110 7135 7150 7820 7885
D4$    - 7150 7820 7885
D5     - 7150 7820 7885
Q0$(   - 7395 7430 8205 8235
Q1     - 7375 7410 7425 7435 7505 8160 8225 8230 8240
Q2$(   - 7225 7230
Q3$(   - 7400 7430 8005 8215 8235
Q4     - 7060 7115 7140
Q7$(   - 8000
Q9$(   - 8005 8010
R0     - 7400 7405 7410 8215 8220 8225
R0$    - 8005
R0$(   - 8180 8185
R1     - 7400 8215
R2$(   - 8185 8190
R4$(   - 7230
R5$(   - 7460 7700 8265
R9$(   - 7440 7445 7455 8245 8250 8260
T1     - 7390 7400 7430 7620 7750 7755 7990 8000 8005 8015 8020 8210 8215 8235
T3     - 7425 7430 7455 7460 7625 7665 7685 7690 7695 7700 7705 8230 8235 8260
         8265
T4     - 7650 7655
T5     - 7445 7450 7455 7660 7665 7685 7690 8250 8255 8260
T6     - 7450 7455 8255 8260
T7     - 7385 7480 7995 8010 8015 8025 8035 8170 8175 8195 8310
T9     - 7090 7370 7485 8315 8320
U0     - 7990 8000
U2$(   - 7355 7390 7415 7440 7495 7655 7970 8010 8015 8150 8210 8245 8330
U3     - 7365 7465 7470 7510 7645 7670 7735 8280 8285
U3$(   - 7355 7360 7415 7465 7495 7650 8150 8185 8280 8330
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 13
       - 7155
7050 DEFFN' 50
???? DEFFN' 100
       - 7455 7460 7750 7815 7880 8260 8265
???? DEFFN' 101
       - 7235
???? DEFFN' 102
       - 7045
???? DEFFN' 105
       - 7400 8215
???? DEFFN' 107
       - 7430 8235
???? DEFFN' 110
       - 7150 7820 7885
???? DEFFN' 200
       - 7385 7645 8170
???? DEFFN' 202
       - 7715 8290
???? DEFFN' 205
       - 7105 7130
???? DEFFN' 210
       - 7070 7145
7215 DEFFN' 211
       - 7075
7350 DEFFN' 212
       - 7080
7600 DEFFN' 213
       - 7085
7810 DEFFN' 214
       - 7095
7875 DEFFN' 215
       - 7125
7965 DEFFN' 216
       - 7065
8150 DEFFN' 217
       - 7415 7495 8330
```

5.23 ATH3Ø6.7

The purpose of the 'ATH3Ø6.7' program is to create the Test Yourself (Flowchart) page of the learning module.

The remainder of this section presents the 'ATH3Ø6.7' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.7 - CNTRL

```
7000 %## Program > AUTHOR/3   (ATH306.7)   Version > 2.0  2200 T/VP  780322
7010 REM **  Abstract  ************************************************************
7020 REM **                                                                        **
7030 REM **   This program will construct the program flowchart                    **
7040 REM **   used to precede the 'Test Yourself' section.                         **
7050 REM **                                                                        **
7060 REM ************************************************************************
7070 REM **
7080 REM **
7090  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.7","9000"): REM LOAD DATA FORAMTS
7100  DEFFN ' 050                                     : REM DEFINE ENTRY
7110              D3 = 32                              : REM CLR PAC INDX FLG
7120              Q4 = 00                              : REM CLR ADRS INDX
7130  GOSUB ' 210                                     : REM INSRT SBL SET #
7140  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAC ON DISK
7150  GOSUB ' 014                                     : REM RTURN TO AUTO MODE
```

AUTHOR System: Phase 3 - ATH306.7 - V MAP

D1 - 7140
D2 - 7140
D3 - 7110 7140
D4$ - 7140
D5 - 7140
G4 - 7120

```
AUTHOR System: Phase 3 - ATH306.7 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 14
       - 7150
7100 DEFFN' 50
???? DEFFN' 102
       - 7090
???? DEFFN' 110
       - 7140
???? DEFFN' 210
       - 7130
```

5.24 ATH3Ø6.8

The purpose of the 'ATH3Ø6.8' program is to create the Test Yourself (Directions, Memory Aids, and Answers) pages of the learning module.

The remainder of this section presents the 'ATH3Ø6.8' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.8 - CNTRL

```
7000 %## Program > AUTHOR/3  (ATH306.8)   Version > 2.0  2200 T/VP  780323
7010 REM **  Abstract  ************************************************
7020 REM **                                                           **
7030 REM **    This program will construct the test and answer frames **
7040 REM **    included in the 'Test Yourself' section.               **
7050 REM **                                                           **
7060 REM ***************************************************************
7070 REM **
7080 REM **
7090  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.8","9000"): REM LOAD DATA FORMATS
7100  DEFFN ' 050                                     : REM DEFINE ENTRY
7110              D3 = 00                             : REM SET PAG INDX FLG
7120              Q4 = 00                             : REM CLR ADRS INDX
7130  GOSUB ' 210                                     : REM INSRT SBL SET #
7140  GOSUB ' 211                                     : REM GEN RND SBL LIST
7150  GOSUB ' 212                                     : REM INSRT TEST SBLS
7160  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
7170 REM **
7180  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
7190              D3 = 32                             : REM CLR PAG INDX FLG
7200              Q4 = 00                             : REM CLR ADRS INDX
7210  GOSUB ' 210                                     : REM INSRT SBL SET #
7220  GOSUB ' 213                                     : REM SAVE LOWR TEXT
7230  GOSUB ' 214 (5)                                 : REM INSRT MEM AIDS
7240  GOSUB ' 215                                     : REM SAVE PAG ON DISK
7250 REM **
7260  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
7270              D3 = 00                             : REM SET PAG INDX FLG
7280              Q4 = 00                             : REM CLR ADRS INDX
7290  GOSUB ' 210                                     : REM INSRT SBL SET #
7300  GOSUB ' 213                                     : REM SAVE LOWR TEXT
7310  GOSUB ' 214 (4)                                 : REM INSRT ANSWERS
7320  GOSUB ' 215                                     : REM SAVE PAGE ON DISK
7330 REM **
7340  GOSUB ' 015                                     : REM RTURN TO AUTO MODE
```

```
7350 REM **
7360 REM **
7370 REM **  Subroutine  ********************************************** 2 1 1
7380 REM **  Abstract  >
7390 REM **    To generate a random list of symbols from the
7400 REM **    current symbol set.
7410 REM **  Variables >
7420 REM **    U2$() > random symbol list      ! B1$() > system memory buffr
7430 REM **    T1 > symbol tbl indx (temp)     ! U0 > symbol tbl indx
7440 REM **    T7 > total symbols in list      ! B8 > random subscript
7450 REM **    U3$() > temp symbol list        ! T2 > sbl lst indx (temp)
7460 REM **    Q7$() > symbol id table         ! Q3$() > symbol lokup table
7470 REM **    Q9$() > i/o buffer              ! R0$ > scratch register
7480 REM **
7490 REM ***********************************************
7500  DEFFN ' 211                                          : REM ENTRY
7510  DIM          U2$(15)16,U3$(15)16                     : REM DEFINE WORK BUFRS
7520  INIT(00)     U2$()                                   : REM CLR TABLE
7530  INIT(00)     U3$()                                   : REM CLR TABLE
7540  INIT(00)     B1$()                                   : REM CLR SYS MEM BUFR
7550               T1 = U0                                 : REM INIT SBL TBL INDX
7560               T2 = 1                                  : REM INIT SBL LST INDX
7570               T7 = 1                                  : REM INIT SBL LST INDX
7580  IF           Q7$(U0,1) <> Q7$(T1,1)    THEN 7650     : REM END OF SBL LIST
7590  DATA LOAD    BA T #2, (Q3$(T1,3),R0$) Q9$()          : REM FETCH SYMBOL
7600  MAT COPY     Q9$()<1,15> TO U3$()<(T7-1)*16+1,15>    : REM STORE SYMBOL
7610               BIN(STR(U3$(T7),16,1)) = T1             : REM STOR SBL TBL INDX
7620               T1 = T1 + 1                             : REM INCR SBL TBL INDX
7630               T7 = T7 + 1                             : REM INCR SBL LST INDX
7640  GOTO         7580                                    : REM CHECK NXT SBL
7650               T7 = T7 - 1                             : REM CMPUT LIST SIZE
7660  GOSUB ' 200 (T7,B1$(1),B8,20)                        : REM GEN RND SUBSCRIPT
7670               U2$(T2) = U3$(B8)                       : REM STOR RND SYMBOL
7680               T2 = T2 + 1                             : REM INCR SBL LST INDX
7690  IF           VAL(B1$(1)) < T7          THEN 7660     : REM NOT END OF LIST
7700  RETURN                                               : REM EXIT
```

```
7710 REM **
7720 REM **
7730 REM **  Subroutine  ********************************************** 2 1 2
7740 REM **  Abstract >
7750 REM **    To insert the list of random test symbols.
7760 REM **  Variables >
7770 REM **    T1$() > item number buffer    ! T2 > page buffer line indx
7780 REM **    T1 > symbol tbl indx          ! U2$() > rnd symbol list
7790 REM **    R4$() > conversion buffer     ! T7 > total symbol in list
7800 REM **    T3 > end of column 1          ! T4 > begining of column 2
7810 REM **
7820 REM ***********************************************
7830  DEFFN ' 212                                          : REM ENTRY
7840  DIM          T1$(2)4                                 : REM DEFINE WORK BUFRS
7850               T2 = 10                                 : REM INIT BUFR LIN INDX
7860               T3 = INT((T7-1)/2) + 1                  : REM CMPUT MID POINT
7870  FOR          T1 = 1 TO T3                            : REM LIST TABLE
7880               T4 = T1 + T3                            : REM CMPUT BEG COL 2
7890  CONVERT      T1   TO T1$(1), (##.#)                  : REM CREATE ITEM ID
7900  CONVERT      T4   TO T1$(2), (##.#)                  : REM CREATE ITEM ID
7910  $TRAN        (T1$()<1,1>, R4$()<185,4>) R            : REM BLANK FILL NUMBR
7920  $TRAN        (T1$()<4,1>, R4$()<185,4>) R            : REM BLANK FILL NUMBR
7930  $TRAN        (T1$()<5,1>, R4$()<185,4>) R            : REM BLANK FILL NUMBR
7940  $TRAN        (T1$()<8,1>, R4$()<185,4>) R            : REM BLANK FILL NUMBR
7950  IF           STR(U2$(T1),16,1)=HEX(00) THEN 7980     : REM NULL SYMBOL
7960  GOSUB ' 100 (T2,24,4,T1$(1),0)                       : REM INSRT ITEM NUMBR
7970  GOSUB ' 100 (T2,28,15,U2$(T1),0)                     : REM INSRT SYMBOL
7980  IF           STR(U2$(T4),16,1)=HEX(00) THEN 8010     : REM NULL SYMBOL
7990  GOSUB ' 100 (T2,44,4,T1$(2),0)                       : REM INSRT ITEM NUMBR
8000  GOSUB ' 100 (T2,48,15,U2$(T4),0)                     : REM INSRT SYMBOL
8010               T2 = T2 + 1                             : REM INCR LINE INDX
8020  NEXT         T1                                      : REM CONT TBL LIST
8030  RETURN                                               : REM EXIT
```

```
8040 REM **
8050 REM **
8060 REM **  Subroutine  ********************************************** 2 1 3
8070 REM **  Abstract >
8080 REM **    To save the last line of page buffer in a temp
8090 REM **    storage buffer.
8100 REM **  Variables >
8110 REM **    BO$() > temp storage buffr  ! Q2$() > output page buffer
8120 REM **    R4$() > character conv bufr !
8130 REM **
8140 REM ***********************************************
8150  DEFFN ' 213                                                : REM ENTRY
8160  DIM          BO$(1)64                                      : REM DEFINE WORK BUFR
8170  MAT COPY     Q2$()<961,64> TO BO$()<1,64>                  : REM LOAD TEMP BUFFR
8180  $TRAN        (Q2$()<961,64>,  R4$()<177,4>)00 R            : REM CLR PAGE BUFFR
8190  GOSUB ' 101 (16,1,1,0)                                     : REM CLR CRT SCREEN
8200  RETURN                                                     : REM EXIT
```

```
8210 REM **
8220 REM **
8230 REM **  Subroutine  ********************************************** 2 1 4
8240 REM **  Abstract >
8250 REM **    To insert either the memory aids or the definitions
8260 REM **    for those symbols used in the test.
8270 REM **  Variables >
8280 REM **    Q1 > output bufr line indx    ! T1 > symbol list indx
8290 REM **    T2 > symbol tbl lokup indx    ! Q0$() > register stack
8300 REM **    Q3$() > sbl lookup table      ! R0 > buffer size
8310 REM **    R1 > item id                  ! Q2$() > output page buffer
8320 REM **    R4$() > char conversion bufr ! T3 > bufr line indx (temp)
8330 REM **    R9$() > scratch line bufr     ! R5$() > single char storage
8340 REM **    U2$() > rnd symbol list       ! T4 > symbol tbl item index
8350 REM **    T7 > total symbols to list    ! T5 > symbol list indx (temp)
8360 REM **
8370 REM ***********************************************
8380  DEFFN ' 214 (T4)                                    : REM ENTRY
8390              Q1 = 3                                  : REM INIT BUFR LIN INDX
8400  FOR         T1 = 1 TO T7                            : REM BEGIN LISTING
8410              T2 = VAL(STR(U2$(T1),16,1))             : REM FETCH SBL TBL INDX
8420              Q0$(1) = "1"                            : REM CLR INDX LOKUP FLG
8430  GOSUB ' 105 (Q3$(T2,14),R0,R1)                      : REM FETCH SBL TEXT
8440              R0 = INT(R0/32) + 1                     : REM CMPUT BUFR SIZE
8450  IF          16-Q1 >= R0                THEN 8510    : REM SPACE AVAIABLE
8460  GOSUB ' 100 (16,41,23,"To Continue Go to :nxt0",0): REM INSRT EOP MSG
8470  GOSUB ' 110 (D1,D2,D3,D4$,D5)                       : REM SAVE PAG ON DISK
8480  GOSUB ' 101 (3,1,14,0)                              : REM CLR CRT SCREEN
8490  $TRAN       (Q2$()<129,896>, R4$()<177,4>)00 R      : REM CLR PAGE BUFFR
8500              Q1 = 3                                  : REM RESET BUFR INDX
8510              T3 = Q1                                 : REM STORE BUFR INDX
8520  GOSUB ' 107 (T3,26,62,Q3$(T2,14),VAL(Q0$(10)))      : REM INSRT TEXT
8530              Q1 = Q1 + 1                             : REM INCR BUFR INDX
8540              T5 = (T1-1)*16 + 1                      : REM CMPUT LST TBL IDX
8550  CONVERT     T1 TO R9$(1), (##.#)                    : REM CREATE ITEM NUMBR
8560  $TRAN       (R9$()<1,1>,  R4$()<185,4>) R           : REM BLANK FILL NUMBER
8570  $TRAN       (R9$()<4,1>,  R4$()<185,4>) R           : REM BLANK FILL NUMBER
8580  MAT COPY    U2$()<T5,15> TO R9$()<5,16>             : REM LOAD SYMBOL
8590  GOSUB ' 100 (T3,6,20,R9$(1),0)                      : REM INSRT SYMBOL
8600  NEXT        T1                                      : REM CONT LISTING
8610  RETURN                                              : REM EXIT
```

```
8620 REM **
8630 REM **
8640 REM **  Subroutine  ********************************************** 2 1 5
8650 REM **  Abstract >
8660 REM **    To save page on disk after reloading last line of
8670 REM **    text from storage buffer.
8680 REM **  Variables >
8690 REM **    BO$() > temp storage buffer  ! Q2$() > output page buffer
8700 REM **
8710 REM ********************************************
8720  DEFFN ' 215                                          : REM ENTRY
8730  GOSUB ' 100 (16,1,64,BO$(1),0)                       : REM INSRT LAST LINE
8740  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAGE ON DISK
8750  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH306.8 - # MAP

7580 - 7640
7650 - 7580
7660 - 7690
7980 - 7950
8010 - 7980
8510 - 8450

```
B0$(   - 8160 8170 8730
B1$(   - 7540 7660 7690
B8     - 7660 7670
D1     - 7160 8470 8740
D2     - 7160 8470 8740
D3     - 7110 7160 7190 7270 8470 8740
D4$    - 7160 8470 8740
D5     - 7160 8470 8740
Q0$(   - 8420 8520
Q1     - 8390 8450 8500 8510 8530
Q2$(   - 8170 8180 8490
Q3$(   - 7590 8430 8520
Q4     - 7120 7200 7280
Q7$(   - 7580
Q9$(   - 7590 7600
R0     - 8430 8440 8450
R0$    - 7590
R1     - 8430
R4$(   - 7910 7920 7930 7940 8180 8490 8560 8570
R9$(   - 8550 8560 8570 8580 8590
T1     - 7550 7580 7590 7610 7620 7870 7880 7890 7950 7970 8020 8400 8410 8540
         8550 8600
T1$(   - 7840 7890 7900 7910 7920 7930 7940 7960 7990
T2     - 7560 7670 7680 7850 7960 7970 7990 8000 8010 8410 8430 8520
T3     - 7860 7870 7880 8510 8520 8590
T4     - 7880 7900 7980 8000 8380 8430 8520
T5     - 8540 8580
T7     - 7570 7600 7610 7630 7650 7660 7690 7860 8400
U0     - 7550 7580
U2$(   - 7510 7520 7670 7950 7970 7980 8000 8410 8580
U3$(   - 7510 7530 7600 7610 7670
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 15
       - 7340
7100 DEFFN' 50
???? DEFFN' 100
       - 7960 7970 7990 8000 8460 8590 8730
???? DEFFN' 101
       - 8190 8480
???? DEFFN' 102
       - 7090
???? DEFFN' 105
       - 8430
???? DEFFN' 107
       - 8520
???? DEFFN' 110
       - 7160 8470 8740
???? DEFFN' 200
       - 7660
???? DEFFN' 205
       - 7180 7260
???? DEFFN' 210
       - 7130 7210 7290
7500 DEFFN' 211
       - 7140
7830 DEFFN' 212
       - 7150
8150 DEFFN' 213
       - 7220 7300
8380 DEFFN' 214
       - 7230 7310
8720 DEFFN' 215
       - 7240 7320
```

5.25 ATH3Ø6.9

The purpose of the 'ATH3Ø6.9' program is to create the Test Yourself (End of Test and Rest Directions) pages of the learning module.

The remainder of this section presents the 'ATH3Ø6.9' program listing and reference maps.

AUTHOR System: Phase 3 - ATH306.9 - CNTRL

```
7000 %## Program > AUTHOR/3   (ATH306.9)   Version > 2.0   2200 T/VP  780323
7010 REM **  Abstract  ************************************************************
7020 REM **                                                                        **
7030 REM **    This program will construct the end of test                         **
7040 REM **    instructions for the 'Self Test'.                                   **
7050 REM **                                                                        **
7060 REM ****************************************************************************
7070 REM **
7080 REM **
7090  DIM          T9$1                                          : REM DEFINE FLAG
7100  GOSUB ' 102 ("DATASUBPROGRAM","ATH106.9","9000"):REM LOAD DATA FORAMTS
7110  DEFFN ' 050                                                : REM DEFINE ENTRY
7120               D3 = 32                                       : REM CLR PAG INDX FLG
7130               Q4 = 00                                       : REM CLR ADRS INDX
7140  GOSUB ' 210                                                : REM INSRT SBL SET #
7150  GOSUB ' 110 (D1,D2,D3,D4$,D5)                              : REM SAVE AGE ON DISK
7160 REM **
7170  GOSUB ' 205 (1)                                            : REM LOAD NXT FORMAT
7180               D3 = 32                                       : REM CLR PAG INDX FLG
7190               Q4 = 00                                       : REM CLR ADRS INDX
7200  GOSUB ' 210                                                : REM INSRT SBL SET #
7210  GOSUB ' 211                                                : REM INSRT SBL SET #
7220  GOSUB ' 212                                                : REM INSRT END OF SET
7230  GOSUB ' 110 (D1,D2,D3,D4$,D5)                              : REM SAVE PAGE ON DISK
7240 REM **
7250  IF           T9$ = HEX(00)              THEN 7270          : REM END OF SBL SETS
7260  GOSUB ' 006                                                : REM RTURN TO BEG SEQU
7270  GOSUB ' 025                                                : REM RTURN TO NXT SEQU
```

```
7280 REM **
7290 REM **
7300 REM **  Subroutine  ********************************************** 2 1 1
7310 REM **  Abstract >
7320 REM **    To insert the current symbol set number into
7330 REM **    the text.
7340 REM **  Variables >
7350 REM **    Q2$() > output page buffer      ! R9$() > temp storage buffer
7360 REM **    R5$() > single char storage     ! R2$() > srch return buffer
7370 REM **    T2 > buffer position indx       ! T3 > symbol set numbr
7380 REM **    Q7$() > symbol id table         !
7390 REM **
7400 REM ***********************************************
7410  DEFFN ' 211                                          : REM ENTRY
7420  MAT COPY     Q2$()<129,64> TO R9$()<1,64>            : REM STOR TEXT LINE
7430  MAT SEARCH   R9$(), = R5$(10) TO R2$()               : REM SRCH FOR INSRT POS
7440               T2 = FNR(VAL(R2$(1))) + 1               : REM CMPUT POSITION
7450               T3 = VAL(Q7$(U0,1))                     : REM FETCH SBL SET #
7460  CONVERT      T3 TO STR(R9$(1),T2,2), (##)            : REM LOAD SBL SET #
7470  GOSUB ' 100 (3,1,64,R9$(1),0)                        : REM INSRT TEXT
7480  RETURN                                               : REM EXIT
```

```
7490 REM **
7500 REM **
7510 REM **  Subroutine  ********************************************** 2 1 2
7520 REM **  Abstract >
7530 REM **    To check if las symbol set module has been created.
7540 REM **  Variables >
7550 REM **    U0 > current symbol tbl indx    ! T1 > symbol tbl indx (temp)
7560 REM **    Q7$() > symbol id table         ! Q0() > numeric register stack
7570 REM **    R0$() > return buffer flag      ! R9$() > scratch line buffer
7580 REM **
7590 REM ********************************************
7600  DEFFN ' 212                                          : REM ENTRY
7610              T1 = U0                                  : REM INIT SBL TBL INDX
7620  IF          Q7$(U0,1) <> Q7$(T1,1)    THEN 7650      : REM END OF SBL SET
7630              T1 = T1 + 1                              : REM INCR SBL TBL INDX
7640  GOTO        7620                                     : REM CHECK NXT SBL
7650  IF          T1-1 >= Q0(2)             THEN 7690      : REM END OF MODULES
7660              U0 = T1                                  : REM RESET SBL TBL INDX
7670              T9$ = HEX(FF)                            : REM CLR END OF SET FLG
7680  GOTO        7740                                     : REM BRANCH TO EXIT
7690              R9$(1) = "of all ## symbols."            : REM INIT LINE TEXT
7700  CONVERT     Q0(2) TO STR(R9$(1),8,2), (##)           : REM LOAD NUMBR OF SBLS
7710  $TRAN       (R9$()<8,1>,  R4$()<185,4>) R            : REM BLANK FILL NUMBR
7720  GOSUB ' 100 (6,24,40,R9$(1),0)                       : REM INSRT TEXT
7730              T9$ = HEX(00)                            : REM SET END OF SET FLG
7740  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH306.9 - # MAP

7270 - 7250
7620 - 7640
7650 - 7620
7690 - 7650
7740 - 7680

```
D1     - 7150 7230
D2     - 7150 7230
D3     - 7120 7150 7180 7230
D4$    - 7150 7230
D5     - 7150 7230
Q0(    - 7650 7700
Q2$(   - 7420
Q4     - 7130 7190
Q7$(   - 7450 7620
R2$(   - 7430 7440
R4$(   - 7710
R5$(   - 7430
R9$(   - 7420 7430 7460 7470 7690 7700 7710 7720
T1     - 7610 7620 7630 7650 7660
T2     - 7440 7460
T3     - 7450 7460
T9$    - 7090 7250 7670 7730
U0     - 7450 7610 7620 7660
```

```
AUTHOR System:  Phase 3 - ATH306.9 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 6
       - 7260
???? DEFFN' 25
       - 7270
7110 DEFFN' 50
???? DEFFN' 100
       - 7470 7720
???? DEFFN' 102
       - 7100
???? DEFFN' 110
       - 7150 7230
???? DEFFN' 205
       - 7170
???? DEFFN' 210
       - 7140 7200
7410 DEFFN' 211
       - 7210
7600 DEFFN' 212
       - 7220
```

5.26 ATH3Ø7.Ø

The purpose of the 'ATH3Ø7.Ø' program is to create the Practice Defining All the Symbols (Overview and Direction) page of the learning module.

The remainder of this section presents the 'ATH3Ø7.Ø' program listing and reference maps.

```
6000 %## Program > AUTHOR/3   (ATH307.0)  Version > 2.0  2200 T/VP  780323
6010 REM **  Abstract  ************************************************************
6020 REM **                                                                        **
6030 REM **    This program will construct the introduction frame                  **
6040 REM **    to the 'Practice Defining All The Symbols' section.                 **
6050 REM **                                                                        **
6060 REM ************************************************************************
6070 REM **
6080 REM **
6090  GOSUB ' 102 ("DATASUBPROGRAM","ATH107.0","9000"): REM LOAD DATA FORMATS
6100  DEFFN ' 050                                     : REM DEFINE ENTRY
6110              D3 = 32                             : REM CLR PAG INDX FLG
6120              Q4 = 44                             : REM SET ADRS INDX
6130  GOSUB ' 210                                     : REM INSRT # OF SBLS
6140  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
6150  GOSUB ' 026                                     : REM RTURN TO AUTO MODE
```

```
6160 REM **
6170 REM **
6180 REM **  Subroutine  ********************************************** 2 1 0
6190 REM **  Abstract >
6200 REM **    To insert the total number of symbols into the
6210 REM **    text of the format.
6220 REM **  Variables >
6230 REM **    Q2$() > ouput page buffer    ! R9$() > scratch line buffer
6240 REM **    R5$() > single char storage ! T1 > buffer position
6250 REM **    Q0() > numeric register stk ! R4$() > conversion buffer
6260 REM **
6270 REM **********************************************
6280  DEFFN ' 210                                          : REM ENTRY
6290  MAT COPY     Q2$()<129,64> TO R9$()<1,64>            : REM LOAD LINE BUFFER
6300  MAT SEARCH   R9$(), = R5$(10) TO R2$()               : REM SRCH LINE BUFR
6310               T1 = FNR(VAL(R2$(1)))                   : REM CMPUT BEG POS
6320  CONVERT      Q0(2) TO STR(R9$(1),T1,2), (##)         : REM LOAD TOTAL
6330  $TRAN        (R9$()<T1,1>, R4$()<185,4>) R           : REM BLANK FILL NUMBR
6340  GOSUB ' 100 (3,1,64,R9$(1),0)                        : REM INSRT TEXT
6350  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH307.0 - V MAP

```
D1     - 6140
D2     - 6140
D3     - 6110 6140
D4$    - 6140
D5     - 6140
Q0(    - 6320
Q2$(   - 6290
Q4     - 6120
R2$(   - 6300 6310.
R4$(   - 6330
R5$(   - 6300
R9$(   - 6290 6300 6320 6330 6340
T1     - 6310 6320 6330
```

```
AUTHOR System: Phase 3 - ATH307.0 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 26
       - 6150
6100 DEFFN' 50
???? DEFFN' 100
       - 6340
???? DEFFN' 102
       - 6090
???? DEFFN' 110
       - 6140
6280 DEFFN' 210
       - 6130
```

5.27 ATH3Ø7.1

The purpose of the 'ATH3Ø7.1' program is to create the Practice Defining All the Symbols (Practice Exercises and Answers) pages of the learning module.

The remainder of this section presents the 'ATH3Ø7.1' program listing and reference maps.

AUTHOR System: Phase 3 - ATH307.1 - CNTRL

```
6000 %## Program > AUTHOR/3  (ATH307.1)  Version > 2.0  2200 T/VP  780323
6010 REM **  Abstract  ************************************************
6020 REM **                                                           **
6030 REM **    This program will construct the practice excersies     **
6040 REM **    and answers for the 'Practice Defining All the Symbols' **
6050 REM **    section.                                               **
6060 REM ***************************************************************
6070 REM **
6080 REM **
6090  GOSUB ' 102 ("DATASUBPROGRAM","ATH107.1","9000"): REM LOAD DATA FORMATS
6100  DEFFN ' 050                                    : REM DEFINE ENTRY
6110 REM **
6120  DIM         T9$1,B2$(20)1                      : REM DEFINE WORK BUFRS
6130  INIT(00)    B2$()                              : REM CLR MEM BUFR #1
6140              D3 = 00                            : REM SET PAG INDX FLG
6150              G0(12) = 01                        : REM SET NUL DISPLY FLG
6160  GOSUB ' 210                                    : REM GEN SBL LST TBL
6170              G0(12) = 00                        : REM CLR NUL DISPLY FLG
6180  RESTORE     1                                  : REM RESTOR DATA POINTR
6190  GOSUB ' 205 (1)                                : REM RESTOR DATA FORMAT
6200              G4 = 00                            : REM CLR ADRS INDX
6210  GOSUB ' 211                                    : REM INSRT SBL LIST
6220  GOSUB ' 110 (D1,D2,D3,D4$,D5)                  : REM SAVE PAG ON DISK
6230 REM **
6240  GOSUB ' 205 (1)                                : REM LOAD NXT FORMAT
6250              D3 = 32                            : REM CLR PAG INDX FLG
6260              G4 = 00                            : REM CLR ADRS INDX
6270  GOSUB ' 212                                    : REM SAVE LAST BUFR LIN
6280  GOSUB ' 213                                    : REM INSRT SBL ANWSRS
6290  GOSUB ' 214                                    : REM SAVE PAG ON DISK
6300 REM **
6310  IF          T9$ = HEX(00)          THEN 6150   : REM CONT EXCERISES
6320  GOSUB ' 027                                    : REM RTURN TO AUTO MODE
```

```
6330 REM **
6340 REM **
6350 REM **  Subroutine  ********************************************** 2 1 0
6360 REM **  Abstract >
6370 REM **    To generate a list of symbols to be used as practice
6380 REM **    symbols in the two frame sequence.
6390 REM **    If insufficient symbols to fill space or symbol
6400 REM **    definition too long fill space with others symbols
6410 REM **    from symbol table not already list on current page.
6420 REM **  Variables >
6430 REM **    U2$() > symbl lst table     ! Q3$() > sbl tbl lokup indx
6440 REM **    B1$() > systm memory buffr  ! B2$() > memory buffr #1
6450 REM **    U2 > sbl lst lokup tbl indx ! T9$ > end of sequence flag
6460 REM **    Q1 > page buffer line indx  ! Q0$() > stack register flags
6470 REM **    R0 > storage bufr size      ! R1 > item id indx
6480 REM **    T3 > bufr line indx (temp)  ! Q9$() > i/o buffer
6490 REM **    Q0() > numeric register stk ! B8 > random subscript
6500 REM **
6510 REM ***********************************************
6520  DEFFN ' 210                                          : REM ENTRY
6530  DIM       B3$(20)1,U2$(15)16                         : REM DEFINE WORK BUFRS
6540 REM **
6550  PRINT     HEX(030A0A0A0A)                            : REM CLR CRT SCREEN
6560  PRINT     " G e n e r a t i n g   S y m b o l";
6570  PRINT     "   L i s t   T a b l e . . ."
6580 REM **
6590  INIT(FF)  U2$()                                      : REM INIT SBL LST BUFR
6600  INIT(00)  B3$()                                      : REM CLR MEM BUFR #2
6610  INIT(00)  B1$()                                      : REM CLR SYS MEM BUFR
6620            U2 = 1                                     : REM INIT LOKUP INDX
6630            T9$ = HEX(00)                              : REM INIT EOF FLAG
6640            Q1 = 3                                     : REM INIT LINE INDX
6650  MAT COPY  B2$() TO B1$()                             : REM RESET SYS MEM BUFR
6660  GOSUB ' 200 (Q0(2),B1$(1),B8,20)                     : REM GEN RND SUBSCRPT
6670            Q0$(1) = "1"                               : REM SET NUL LOKUP FLG
6680  GOSUB ' 105 (Q3$(B8,4),R0,R1)                        : REM FETCH DEFINITION
6690            R0 = INT(R0/36) + 1                        : REM CMPUT APPROX SIZ
6700  IF        16-Q1 >= R0                 THEN 6730      : REM IF ENOUGH SPACE
6710  GOSUB ' 215 (U2$(1),B2$(1))                          : REM FILL REMAING SPACE
6720  GOTO      6870                                       : REM BRANCH TO EXIT
6730            T3 = Q1                                    : REM STORE LINE INDX
6740  GOSUB ' 107 (T3,26,62,Q3$(B8,4),VAL(Q0$(10)))        : REM INSRT DEFINITION
6750            Q1 = Q1 + 1                                : REM INCR LINE INDX
6760  DATA LOAD BA T #2, (Q3$(B8,3),R0$) Q9$()             : REM FETCH SYMBOL
6770  MAT COPY  Q9$()<1,15> TO U2$()<(U2-1)*16+1,15>       : REM STORE SYMBOL
6780            BIN(STR(U2$(U2),16,1)) = B8                : REM STORE LOOKUP INDX
6790            U2 = U2 + 1                                : REM INCR LST TBL IDX
6800  MAT COPY  B1$() TO B2$()                             : REM UPDAT MEM TBL #1
6810  IF        Q0(2) <> VAL(B2$(1))        THEN 6860      : REM IF END OF SEQU
6820            T9$ = HEX(FF)                              : REM SET EOF FLAG
6830  INIT(00)  B2$()                                      : REM CLR MEM TBL #1
6840  GOSUB ' 215 (U2$(1),B2$(1))                          : REM FIL REMAING SPACE
6850  GOTO      6870                                       : REM BRANCH TO EXIT
6860  IF        Q1 < 16                     THEN 6660      : REM FETCH NEW SBL
6870            U2 = U2 - 1                                : REM CMPUT TOTAL SBLS
6880  RETURN                                               : REM EXIT
```

```
6890 REM **
6900 REM **
6910 REM **  Subroutine  ********************************************** 2 1 1
6920 REM **  Abstract >
6930 REM **    To create a list of symbols selected from those symbols
6940 REM **    to be used on page 2 of sequence. The list is 14 lines
6950 REM **    of randomly ordered symbols.
6960 REM **  Variables >
6970 REM **    B4$() > temp line buffer       ! B5$() > curnt symbol buffer
6980 REM **    B1$() > system memory buffer   ! B3$() > memory buffer #2
6990 REM **    G0$() > stack register buffer  ! U2 > total symbls in list
7000 REM **    ?? > random subscript          ! T1 > bufr line indx (loop)
7010 REM **    T3 > temp line bufr index      ! T4 > sbl list lokup index
7020 REM **    T5 > length of symbol          ! R5$() > single char buffer
7030 REM **
7040 REM ******************************************
7050  DEFFN ' 211                                          : REM ENTRY
7060  DIM         B4$(1)64,B5$(1)15                        : REM DEFINE WORK BUFRS
7070  INIT(00)    B1$()                                    : REM CLR SYS MEM BUFR
7080  INIT(00)    B3$()                                    : REM CLR MEM BUFR #2
7090  FOR         T1 = 2 TO 15                             : REM BEGIN LINE INDX
7100              T3 = 24                                  : REM INIT COL INDX
7110  INIT(20)    B4$()                                    : REM CLR LINE BUFR
7120  MAT COPY    B3$() TO B1$()                           : REM RESET SYS MEM BUFR
7130  MAT COPY    B1$() TO B3$()                           : REM UPDAT MEM BUFR #1
7140  GOSUB ' 200 (U2,B1$(1),88,20)                        : REM GEN RND SBSCRPT
7150              T4 = 88                                  : REM FETCH TBL INDX
7160  MAT COPY    U2$()<(T4-1)*16+1,15> TO B5$()<1,15>     : REM FETCH SYMBOL
7170              T5 = LEN(B5$(1))                         : REM CMPUT SBL LENGTH
7180  IF          T3+T5 <= 60                THEN 7220     : REM IF LINE NOT FULL
7190  IF          U2 <> VAL(B1$(1))          THEN 7140     : REM IF LIST NOT EMPTY
7200  INIT(00)    B1$()                                    : REM CLR SYS MEM BUFR
7210  GOTO        7350                                     : REM WRITE CURNT LINE
7220  MAT COPY    B5$()<1,T5> TO B4$()<T3,T5>              : REM LOAD LINE BUFR
7230              T3 = T3 + T5                             : REM INCR LINE INDX
7240  IF          T3 > 56                    THEN 7350     : REM IF LINE FULL
7250  MAT COPY    R5$()<9,1> TO B4$()<T3+1,2>              : REM LOAD SEP CHAR
7260              T3 = T3 + 3                              : REM INCR LINE INDX
7270  MAT COPY    B3$() TO B1$()                           : REM RESET SYS MEM BUFR
7280  GOSUB ' 202 (B1$,B1$(1),88)                          : REM UPDAT SYS MEM BUFR
7290  IF          B1$ <> HEX(00)             THEN 7320     : REM DUPLICIAT SBL
7300              BIN(B1$(1)) = VAL(B1$(1)) + 1            : REM INCR BITS ON COUNT
7310  MAT COPY    B1$() TO B3$()                           : REM UPDAT MEM BUFR #1
7320  IF          U2 <> VAL(B3$(1))          THEN 7130     : REM IF NOT END OF LIST
7330  INIT(00)    B1$()                                    : REM CLR SYS MEM BUFR
7340  GOTO        7130                                     : REM START LIST OVER
7350  GOSUB ' 100 (T1,24,40,STR(B4$(1),24,40),0)           : REM WRITE LINE BUFR
7360  NEXT        T1                                       : REM INCR LINE INDX
7370  RETURN                                               : REM EXIT
```

```
7380 REM **
7390 REM **
7400 REM **  Subroutine  ********************************************** 2 1 2
7410 REM **  Abstract >
7420 REM **    To save last line of text buffer in a temp
7430 REM **    storage buffer.
7440 REM **  Variables >
7450 REM **    Q2$() > page buffer          ! B0$() > temp storage buffer
7460 REM **    R4$() > conversion buffer    !
7470 REM **
7480 REM ***********************************************
7490  DEFFN ' 212                                          : REM ENTRY
7500  DIM         B0$(2)64                                 : REM DEFINE WORK BUFRS
7510  MAT COPY    Q2$()<961,64> TO B0$()<1,64>             : REM LOAD WRK BUFR
7520              STR(B0$(2),41)="To Continue Go to :nxt0": REM LOAD WRK BUFR
7530  $TRAN       (Q2$()<961,64> , R4$()<177,4>)00 R       : REM CLR TEXT BUFFER
7540  GOSUB ' 101 (16,1,1,0)                               : REM CLR CRT SCREEN
7550  RETURN                                               : REM EXIT
```

```
7560 REM **
7570 REM **
7580 REM **  Subroutine  ********************************************** 2 1 3
7590 REM **  Abstract >
7600 REM **    To insert the answers to the practice excersises
7610 REM **    in a sorted manner on the second frame of sequence.
7620 REM **  Variables >
7630 REM **    U3$() > sort work buffer    ! U4$() > sort locator buffer
7640 REM **    U5$() > sorted symbol list  ! U2$() > symbol list table
7650 REM **    Q1 > output page line indx  ! T1 > sbl list indx (loop)
7660 REM **    T2 > sbl table lookup indx  ! T3 > line indx (temp)
7670 REM **    Q3$() > sbl tbl indx buffr  ! R9$() > scratch buffer
7680 REM **    T5 > symbol length          ! T6 > column center
7690 REM **    R5$() > single char buffer  !
7700 REM **
7710 REM ********************************************
7720  DEFFN ' 213                                          : REM ENTRY
7730  DIM          U3$(15)2,U4$(15)2,U5$(15)16             : REM DEFINE WORK BUFRS
7740  MAT SORT     U2$()          TO U3$(), U4$()          : REM SORT SBL LST
7750  MAT MOVE     U2$(), U4$(1) TO U5$(1)                 : REM GEN SORTED LIST
7760 REM **
7770               Q1 = 3                                  : REM INIT LINE INDX
7780  FOR          T1 = 1 TO U2                            : REM GEN LIST
7790               T2 = VAL(STR(U5$(T1),16,1))             : REM FETCH LOOKUP INDX
7800               Q0$(1) = "1"                            : REM SET NUL LOKUP FLG
7810  GOSUB ' 105 (Q3$(T2,4),R0,R1)                        : REM FETCH DEFINITION
7820               T3 = Q1                                 : REM STOR LINE INDX
7830  GOSUB ' 107 (T3,26,62,Q3$(T2,4),VAL(Q0$(10)))        : REM INSRT DEFINITION
7840               Q1 = Q1 + 1                             : REM INCR LINE INDX
7850               R9$(1) = STR(U5$(T1),1,15)              : REM STORE SYMBOL
7860               T5 = LEN(R9$(1))                        : REM CMPUT SBL LENGTH
7870               T6 = 13 - INT(T5/2)                     : REM CMPUT COL CENTER
7880  GOSUB ' 100 (T3,T6,T5,R9$(1),0)                      : REM INSRT SYMBOL
7890  GOSUB ' 100 (T3,22,1,R5$(9),0)                       : REM INSRT ITEM FLAG
7900  NEXT         T1                                      : REM CONT LISTING
7910  RETURN                                               : REM EXIT
```

```
7920 REM **
7930 REM **
7940 REM **  Subroutine  ********************************************** 2 1 4
7950 REM **  Abstract >
7960 REM **    To save page current page on disk and insert
7970 REM **    message indicating that section continues on next
7980 REM **    page.
7990 REM **
8000 REM ************************************************
8010  DEFFN ' 214                                          : REM ENTRY
8020  IF          T9$ = HEX(00)              THEN 8040 : REM INSRT LAST LINE
8030  GOSUB ' 100 (16,1,64,B0$(1),0):        GOTO 8050 : REM INSRT LAST LINE
8040  GOSUB ' 100 (16,1,64,B0$(2),0):        GOTO 8050 : REM INSRT LAST LINE
8050  GOSUB ' 110 (D1,D2,D3,D4$,D5)                    : REM SAVE PAG ON DSK
8060  RETURN                                           : REM EXIT
```

```
8070 REM **
8080 REM **
8090 REM **, Subroutine  ********************************************** 2 1 5
8100 REM **  Abstract >
8110 REM **    To fill the space remaining on a page with a
8120 REM **    symbol from current set that has not been inserted
8130 REM **    on current page.
8140 REM **  Variables >
8150 REM **    U2$() > symbol list table    ! Q0() > numeric register stack
8160 REM **    B2$() > memory buffer #1     ! B3$() > memory buffer #2
8170 REM **    B1$() > system memory bufr   ! Q1 > output page line indx
8180 REM **    U2 > total sbls in list      ! B8 > random subscript
8190 REM **    R0$() > search id            ! R2$() > srch return buffer
8200 REM **    Q0$() > stack register bufr  ! T1 > sbl tbl lokup indx
8210 REM **    Q3$() > sbl tbl lokup table  ! R0 > buffer size
8220 REM **    R1 > item id                 ! T3 > temp bufr line indx
8230 REM **    Q9$() > i/o buffer           ! R5$() > single char bufr
8240 REM **    B1$ > memory indx flg        ! T9$ > end of list flag
8250 REM **
8260 REM ********************************************
8270  DEFFN ' 215 (U2$(1),B2$(1))                          : REM ENTRY
8280  INIT(00)    B3$()                                    : REM CLR MEM BUFR #2
8290  IF          Q1 >= 16                    THEN 8590    : REM IF PAGE FULL
8300  MAT COPY    B3$() TO B1$()                           : REM RESET SYS MEM BUFR
8310  GOSUB ' 200 (Q0(2),B1$(1),B8,20)                     : REM GEN RND SBSCRPT
8320  IF          Q0(2) < VAL(B1$(1))         THEN 8590    : REM END OF LIST
8330              BIN(R0$(1)) = B8                         : REM CONV TO SRCH ID
8340  MAT SEARCH  U2$()<16,225>,=R0$(1)TO R2$()STEP 16:    : REM SRCH FOR SAM SBL
8350  IF          R2$(1) = HEX(0000)          THEN 8380    : REM IF NO DUPLICIATES
8360  IF          U2 <> VAL(B1$(1))           THEN 8310    : REM IF ALL SBLS ON PAG
8370  GOTO        8590                                     : REM BRANCH TO EXIT
8380              Q0$(1) = "1"                             : REM SET NULL LOKUP FLG
8390  GOSUB ' 105 (Q3$(B8,4),R0,R1)                        : REM FETCH DEFINITION
8400              R0 = INT(R0/36) + 1                      : REM CMPUT APPROX SIZ
8410  IF          16-Q1 < R0                  THEN 8310    : REM SBL DOES NOT FIT
8420              T3 = Q1                                  : REM STORE LINE INDX
8430  GOSUB ' 107 (T3,26,62,Q3$(B8,4),VAL(Q0$(10)))        : REM INSRT DEFINITION
8440              Q1 = Q1 + 1                              : REM INCR LINE INDX
8450  DATA LOAD   BA T #2, (Q3$(B8,3),R0$) Q9$()           : REM FETCH SYMBOL
8460  MAT COPY    Q9$()<1,15> TO U2$()<(U2-1)*16+1,15>:    : REM STORE SYMBOL
8470              BIN(STR(U2$(U2),16,1)) = B8              : REM STORE LOOKUP INDX
8480              U2 = U2 + 1                              : REM INCR LST TBL IDX
8490  MAT COPY    B1$() TO B3$()                           : REM UPDAT MEM BUFR #2
8500  MAT COPY    B2$() TO B1$()                           : REM RESET SYS MEM BUFR
8510  GOSUB ' 202 (B1$,B1$(1),B8)                          : REM ADD TO MEM BUFFR
8520  IF          B1$ <> HEX(00)              THEN 8300    : REM DUPLICATE SBL
8530              BIN(B1$(1)) = VAL(B1$(1)) + 1            : REM INCR SYS COUNTER
8540  MAT COPY    B1$() TO B2$()                           : REM UPDAT MEM BUFR #1
8550  IF          Q0(2) <> VAL(B2$(1))        THEN 8300    : REM MORE SBLS TO SCAN
8560              T9$ = HEX(FF)                            : REM SET END OF PAG FLG
8570  INIT(00)    B2$()                                    : REM CLR MEM BUFR #1
8580  GOSUB ' 215 (U2$(1),B2$(1))                          : REM FILL REMAING SPACE
8590  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH307.1 - # MAP

```
6150  - 6310
6660  - 6860
6730  - 6700
6860  - 6810
6870  - 6720 6850
7130  - 7320 7340
7140  - 7190
7220  - 7180
7320  - 7290
7350  - 7210 7240
8040  - 8020
8050  - 8030 8040
8300  - 8520 8550
8310  - 8360 8410
8380  - 8350
8590  - 8290 8320 8370
```

AUTHOR System: Phase 3 - ATH307.1 - V MAP

```
B0$(  - 7500 7510 7520 8030 8040
B1$   - 7280 7290 8510 8520
B1$(  - 6610 6650 6660 6800 7070 7120 7130 7140 7190 7200 7270 7280 7300 7310
        7330 8300 8310 8320 8360 8490 8500 8510 8530 8540
B2$(  - 6120 6130 6650 6710 6800 6810 6830 6840 8270 8500 8540 8550 8570 8580
B3$(  - 6530 6600 7080 7120 7130 7270 7310 7320 8280 8300 8490
B4$(  - 7060 7110 7220 7250 7350
B5$(  - 7060 7160 7170 7220
B8    - 6660 6680 6740 6760 6780 7140 7150 7280 8310 8330 8390 8430 8450 8470
        8510
D1    - 6220 8050
D2    - 6220 8050
D3    - 6140 6220 6250 8050
D4$   - 6220 8050
D5    - 6220 8050
Q0(   - 6150 6170 6660 6810 8310 8320 8550
Q0$(  - 6670 6740 7800 7830 8380 8430
Q1    - 6640 6700 6730 6750 6860 7770 7820 7840 8290 8410 8420 8440
Q2$(  - 7510 7530
Q3$(  - 6680 6740 6760 7810 7830 8390 8430 8450
Q4    - 6200 6260
Q9$(  - 6760 6770 8450 8460
R0    - 6680 6690 6700 7810 8390 8400 8410
R0$   - 6760 8450
R0$(  - 8330 8340
R1    - 6680 7810 8390
R2$(  - 8340 8350
R4$(  - 7530
R5$(  - 7250 7890
R9$(  - 7850 7860 7880
T1    - 7090 7350 7360 7780 7790 7850 7900
T2    - 7790 7810 7830
T3    - 6730 6740 7100 7180 7220 7230 7240 7250 7260 7820 7830 7880 7890 8420
        8430
T4    - 7150 7160
T5    - 7170 7180 7220 7230 7860 7870 7880
T6    - 7870 7880
T9$   - 6120 6310 6630 6820 8020 8560
U2    - 6620 6770 6780 6790 6870 7140 7190 7320 7780 8360 8460 8470 8480
U2$(  - 6530 6590 6710 6770 6780 6840 7160 7740 7750 8270 8340 8460 8470 8580
U3$(  - 7730 7740
U4$(  - 7730 7740 7750
U5$(  - 7730 7750 7790 7850
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 27
       - 6320
6100 DEFFN' 50
???? DEFFN' 100
       - 7350 7880 7890 8030 8040
???? DEFFN' 101
       - 7540
???? DEFFN' 102
       - 6090
???? DEFFN' 105
       - 6680 7810 8390
???? DEFFN' 107
       - 6740 7830 8430
???? DEFFN' 110
       - 6220 8050
???? DEFFN' 200
       - 6660 7140 8310
???? DEFFN' 202
       - 7280 8510
???? DEFFN' 205
       - 6190 6240
6520 DEFFN' 210
       - 6160
7050 DEFFN' 211
       - 6210
7490 DEFFN' 212
       - 6270
7720 DEFFN' 213
       - 6280
8010 DEFFN' 214
       - 6290
8270 DEFFN' 215
       - 6710 6840 8580
```

5.28 ATH3Ø7.2

The purpose of the 'ATH3Ø7.2' program is to create the Practice Defining All the Symbols (Ending and Practice Directions) pages of the learning module.

The remainder of this section presents the 'ATH3Ø7.2' program listing and reference maps.

```
6000 %## Program > AUTHOR/3   (ATH307.2)   Version > 2.0  2200 T/VP  780323
6010 REM **  Abstract  ************************************************************
6020 REM **                                                                        **
6030 REM **    This program will construct the end of practice                     **
6040 REM **    instructions for the 'Practice Defining All The Symbols'            **
6050 REM **    section.                                                            **
6060 REM **                                                                        **
6070 REM ****************************************************************************
6080 REM **
6090 REM **
6100  GOSUB ' 102 ("DATASUBPROGRAM","ATH107.2","9000"): REM LOAD DATA FORMATS
6110  DEFFN ' 050                                     : REM DEFINE ENTRY
6120              D3 = 32                             : REM CLR PAG INDX FLG
6130              Q4 = 00                             : REM CLR ADRS INDX
6140  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
6150 REM **
6160  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
6170              D3 = 32                             : REM CLR PAG INDX FLG
6180              Q4 = 00                             : REM CLR ADRS INDX
6190  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
6200  GOSUB ' 028                                     : REM RTURN TO AUTO MODE
```

AUTHOR System: Phase 3 - ATH307.2 - V MAP

```
D1     - 6140 6190
D2     - 6140 6190
D3     - 6120 6140 6170 6190
D4$    - 6140 6190
D5     - 6140 6190
Q4     - 6130 6180
```

AUTHOR System: Phase 3 - ATH307.2 - ' MAP

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 28
       - 6200
6110 DEFFN' 50
???? DEFFN' 102
       - 6100
???? DEFFN' 110
       - 6140 6190
???? DEFFN' 205
       - 6160
```

5.29 ATH3Ø8.Ø

The purpose of the 'ATH3Ø8.Ø' program is to create the Criterion Test (Directions) page of the learning module.

The remainder of this section presents the 'ATH3Ø8.Ø' program listing and reference maps.

```
6000 %## Program > AUTHOR/3   (ATH308.0)   Version > 2.0  2200 T/VP  780329
6010 REM **  Abstract  ************************************************************
6020 REM **                                                                        **
6030 REM **    This program will construct the introduction frame                  **
6040 REM **    to the 'Criterion Test' section.                                    **
6050 REM **                                                                        **
6060 REM ************************************************************************
6070 REM **
6080 REM **
6090              Q4$(45) = Q4$(42)                        : REM INIT MEMORY ADRS
6100  IF          Q0(11) = 00                THEN 6180 : REM IF NO TESTS
6110 REM **
6120  GOSUB ' 102 ("DATASUBPROGRAM","ATH108.0","9000"): REM LOAD DATA FORMATS
6130  DEFFN ' 050                                      : REM DEFINE ENTRY
6140              D3 = 00                              : REM SET PAG INDX FLG
6150              Q4 = 45                              : REM SET ADRS INDX
6160  GOSUB ' 110 (D1,D2,D3,D4$,D5)                    : REM SAVE PAG ON DISK
6170 REM **
6180  GOSUB ' 029                                      : REM RTURN TO AUTO MODE
```

AUTHOR System: Phase 3 - ATH308.0 - # MAP

6180 - 6100

AUTHOR System: Phase 3 - ATH308.0 - V MAP

```
D1      - 6160
D2      - 6160
D3      - 6140 6160
D4$     - 6160
D5      - 6160
Q0(     - 6100
Q4      - 6150
Q4$(    - 6090
```

```
AUTHOR System: Phase 3 - ATH308.0 -- ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 29
       - 6180
6130 DEFFN' 50
???? DEFFN' 102
       - 6120
???? DEFFN' 110
       - 6160
```

5.30 ATH3Ø8.1

The purpose of the 'ATH3Ø8.1' program is to create the Criterion Test (Tests, Answers, End of Test Directions, and Congratulations) pages of the learning module.

The remainder of this section presents the 'ATH3Ø8.1' program listing and reference maps.

```
6000 %## Program > AUTHOR/3   (ATH308.1)   Version > 2.0  2200 T/VP  780323
6010 REM **  Abstract  ************************************************************
6020 REM **                                                                        **
6030 REM **    This program will construct the desired number of                   **
6040 REM **    criterion tests.  This sequence includes the the test               **
6050 REM **    symbols displayed in a random order, the answers to the             **
6060 REM **    test and, end of test instructions for each test.                   **
6070 REM **                                                                        **
6080 REM ************************************************************************
6090 REM **
6100 REM **
6110  GOSUB ' 102 ("DATASUBPROGRAM","ATH108.1","9000"): REM LOAD DATA FORMATS
6120  DEFFN ' 050                                     : REM DEFINE ENTRY
6130              D3 = 32                             : REM CLR PAG INDX FLG
6140              Q4 = 00                             : REM CLR ADRS INDX
6150              T1 = 00                             : REM INIT TST INDEX
6160 REM **
6170  IF          Q0(11) = 00              THEN 6530 : REM IF NO TESTS
6180 REM **
6190  GOSUB ' 210                                     : REM INSRT TEST NUMBR
6200  GOSUB ' 214                                     : REM SAVE LOWR TEXT
6210  GOSUB ' 211                                     : REM CREATE TEST SBLS
6220  GOSUB ' 216                                     : REM SAVE PAG ON DISK
6230 REM **
6240  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
6250              D3 = 32                             : REM CLR PAG INDX FLG
6260              Q4 = 00                             : REM CLR ADRS INDX
6270  GOSUB ' 210                                     : REM INSRT TEST NUMBR
6280  GOSUB ' 214                                     : REM SAVE LOWR TEXT
6290  GOSUB ' 212                                     : REM CREATE ANSWERS
6300  GOSUB ' 216                                     : REM SAVE PAG ON DISK
6310 REM **
6320  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
6330              D3 = 32                             : REM CLR PAG INDX FLG
6340              Q4 = 00                             : REM CLR ADRS INDX
6350  GOSUB ' 210                                     : REM INSRT TEST NUMBR
6360  GOSUB ' 213                                     : REM CLEAR EXTR TEXT
6370  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
6380 REM **
6390  IF          T1 = Q0(11)              THEN 6460 : REM IF END OF TESTS
6400  RESTORE     1                                   : REM RESTOR DATA PONTR
6410  GOSUB ' 205 (1)                                 : REM RESTOR DATA FMT
6420              D3 = 32                             : REM CLR PAG INDX FLG
6430              Q4 = 00                             : REM CLR ADRS INDX
6440  GOTO        6190                                : REM CREATE NXT TEST
6450 REM **
6460  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
6470              D3 = 32                             : REM CLR PAG INDX FLG
6480              Q4 = 48                             : REM CLR ADRS INDX
6490  GOSUB ' 214                                     : REM SAVE LAST LINE
6500  GOSUB ' 215                                     : REM INSRT GROUP NAME
6510  GOSUB ' 216                                     : REM SAVE PAG ON DISK
6520 REM **
6530  GOSUB ' 030                                     : REM RTURN TO AUTO MODE
```

```
6540 REM **
6550 REM **
6560 REM **  Subroutine  ********************************************** 2 1 0
6570 REM **  Abstract >
6580 REM **    To insert the current criterion test number into
6590 REM **    the frame text.
6600 REM **  Variables >
6610 REM **    Q2$() > output page buffer    ! R2$() > srch return buffer
6620 REM **    R9$() > scratch line bufr     ! T1    > total tests generated
6630 REM **    R5$() > single char storage   ! T2    > line bufr position
6640 REM **
6650 REM ***********************************************
6660  DEFFN ' 210                                          : REM ENTRY
6670  MAT COPY     Q2$()<1,64> TO R9$()<1,64>              : REM LOAD LINE BUFR
6680  ADD          (R9$(1),80)                             : REM CLR UNDERSCORE
6690  MAT SEARCH   R9$(), = R5$(10) TO R2$()               : REM SRCH FOR POS
6700               T2 = FNR(VAL(R2$(1))) + 1               : REM CMPUT POSITION
6710  CONVERT      T1+1 TO STR(R9$(1),T2,2), (##)          : REM LOAD TEST NUMBR
6720  ADD          (R9$(1),80)                             : REM RESET UNDRSCORE
6730  GOSUB ' 100 (1,1,64,R9$(1),0)                        : REM INSRT TEXT
6740  RETURN                                               : REM EXIT
```

```
6750 REM **
6760 REM **
6770 REM **  Subroutine  ********************************************  2 1 1
6780 REM **  Abstract >
6790 REM **    To create criterion Test Questions.
6800 REM **  Variables >
6810 REM **    T1$() > memory sbl ordr  ! Q9$() > scratch bufr
6820 REM **    T5 > total sbl indx      ! T2 > curnt row indx
6830 REM **    T3 > curnt col indx      ! B8 > random subscrpt
6840 REM **    Q3$() > symbol table     ! R9$() > temp lin buf
6850 REM **    R5$() > single char bufr ! R4$() > conversions
6860 REM **    Q0$() > single byte flgs ! Q0() > numeric flags
6870 REM **    T4 > column indicator    !
6880 REM **
6890 REM ***********************************************
6900  DEFFN ' 211                                          : REM ENTRY
6910  DIM           T1$(75)16                              : REM DEFINE BUFFER
6920  INIT(00)      B1$()                                  : REM ZERO MEM TABLE
6930                Q0$(4) = HEX(31)                       : REM INIT COL INDX FLG
6940                T2 = 2                                 : REM ROW INDX
6950                T3 = 6                                 : REM COL INDX
6960  FOR           T5 = 1 TO Q0(2)                        : REM BEGIN SBL TBL LOOP
6970  INIT(SF)      R9$()                                  : REM INIT LINE BUFR
6980  GOSUB ' 200   (Q0(2),B1$(1),B8,20)                   : REM FECTH RNDM SBSCRPT
6990  DATA LOAD     BA T #2, (Q3$(B8,3),K0$)Q9$()          : REM LOAD SYMBOL
7000  MAT COPY      R5$()<1,1> TO Q9$()<16,1>              : REM CLR EOI CHAR
7010  MAT COPY      Q9$()<1,15> TO T1$()<(T5-1)*16+1,16>   : REM STOR SYMBOL
7020                T4 = LEN(T1$(T5))                      : REM CMPUT SBL LEN
7030                BIN(STR(T1$(T5),16,1)) = B8            : REM REMEMBER SEQUENCE
7040  CONVERT       T5 TO STR(R9$(1),1,2), (##)            : REM CMPUTE ITEM NUMBR
7050  $TRAN         (R9$()<1,1>, R4$()<185,4>) R           : REM BLANK FILL NUMBR
7060  MAT COPY      R5$()<8,1> TO R9$()<3,2>               : REM LOAD DECIMAL POINT
7070  MAT COPY      Q9$()<1,T4> TO R9$()<5,T4+2>           : REM LOAD SYMBOL
7080  GOSUB ' 100   (T2,T3,25,R9$(1),0)                    : REM LOAD INTO PAC BUFR
7090                T2 = T2 + 1                            : REM INCR ROW INDX
7100  IF            T2 < 16                  THEN 7220     : REM NOT END OF PAG
7110  IF            Q0$(4) <> "2"            THEN 7180     : REM NOT COL 2
7120  IF            T5 = Q0(2)               THEN 7220     : REM END OF SBL TBL
7130  GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):    REM LOAD CONT MSG
7140  GOSUB ' 110   (D1,D2,D3,D4$,D5)                      : REM SAVE CURRNT PAGE
7150  GOSUB ' 101   (2,1,15,0)                             : REM CLR CRT SCREEN
7160  $TRAN         (Q2$()<65,960>, R4$()<177,4>)00 R      : REM CLR PAGE BUFR
7170                Q0$(4) = "0"                           : REM RESET COL INCTOR
7180  ADD           (Q0$(4),01)                            : REM INCR COL INCTOR
7190                T2 = 2                                 : REM RESET ROW INDX
7200  CONVERT       Q0$(4) TO T4                           : REM CONV COL NUMBR
7210                T3 = 6↑T4                              : REM CMPUTE COL INDX
7220  NEXT          T5                                     : REM CONT SBL TBL LOOP
7230  RETURN                                               : REM EXIT
```

```
7240  REM **
7250  REM **
7260 REM **  Subroutine  ********************************************** 2 1 2
7270 REM **  Abstract >
7280 REM **    To create the Answers for Criterion Test.
7290 REM **  Variables >
7300 REM **    T2 > curnt row indx  ! T5 > sbl sequence indx
7310 REM **    T4 > symbol tbl indx ! T1$() > seqnce mem tbl
7320 REM **    Q3$() > symbol table ! Q0() > numeric flags
7330 REM **    R0 > length of ans.  ! R9$() > temp line bufr
7340 REM **    Q9$() > i/o buffer   ! R4$() > conversions
7350 REM **    R5$() > single chars ! Q1 > output bufr lin indx
7360 REM **
7370 REM ***********************************************
7380  DEFFN ' 212                                          : REM ENTRY
7390                T2 = 3                                 : REM SET ROW INDX
7400  FOR           T5 = 1 TO Q0(2)                        : REM BEG SBL TBL LOOP
7410                T4 = VAL(STR(T1$(T5),16,1))            : REM CMPUT SBL TBL INDX
7420                Q0$(1) = "1"                           : REM SET NULL LOOKUP
7430  GOSUB ' 105   (Q3$(T4,4),R0,R1)                      : REM LOAD DEFINITION
7440                R0 = INT(R0/32) + 1                    : REM CMPUT LINE NEEDED
7450  IF            R0 <= 16-T2                THEN 7510   : REM SPACE AVAIABLE
7460  GOSUB ' 100(16,41,23,"To Continue Go to :nxt0",0):   REM LOAD CONT MSG
7470  GOSUB ' 110   (D1,D2,D3,D4$,D5)                      : REM SAVE PAGE
7480  GOSUB ' 101   (3,1,14,0)                             : REM CLR CRT SCREEN
7490  $TRAN         (Q2$()<129,896>, R4$()<177,4>)00 R     : REM CLR PAGE BUFR
7500                T2 = 3                                 : REM SET ROW INDX
7510  GOSUB ' 107   (T2,28,60,Q3$(T4,4),VAL(Q0$(10)))      : REM LOAD TEXT
7520  INIT(20)      R9$()                                  : REM CLR LINE BUFR
7530  CONVERT       T5 TO STR(R9$(1),1,2), (##)            : REM CMPUT ITEM NUMBR
7540  $TRAN         (R9$()<1,1>, R4$()<185,4>) R           : REM ZERO FILL NUMBR
7550  MAT COPY      R5$()<8,1> TO R9$()<3,2>               : REM LOAD DECIMAL POINT
7560  MAT COPY      T1$()<(T5-1)*16+1,15> TO R9$()<5,16>:  REM LOAD SYMBOL
7570  MAT COPY      R5$()<9,1> TO R9$()<21,2>              : REM LOAD SEPERADR
7580  GOSUB ' 100   (T2,6,22,R9$(1),0)                     : REM LOAD INTO PAC BUFR
7590                T2 = Q1 + 1                            : REM INCR ROW INDX
7600  NEXT          T5                                     : REM CONT SBL TBL LOOP
7610  RETURN                                               : REM EXIT
```

```
7620 REM **
7630 REM **
7640 REM **  Subroutine  ********************************************** 2 1 3
7650 REM **  Abstract >
7660 REM **    To erase and position text to refelect proper
7670 REM **    instructions.
7680 REM **  Variables >
7690 REM **    T1 > total tests generated  ! G0() > numeric register stack
7700 REM **    Q2$() > output page bufr    ! R4$() > conversion buffer
7710 REM **    B0$() > temp storage buffer !
7720 REM **
7730 REM ***********************************************
7740  DEFFN ' 213                                          : REM ENTRY
7750  DIM          B0$(5)64                                : REM DEFINE WORK BUFR
7760               T1 = T1 + 1                             : REM INCR TEST INDX
7770  IF           T1 = G0(11)              THEN 7830      : REM IF END OF TESTS
7780 REM **
7790  GOSUB ' 101 (13,1,3,0)                               : REM CLR CRT SCREEN
7800  $TRAN       (Q2$()<769,192>,  R4$()<177,4>)00 R      : REM CLR PAG BUFFR
7810  GOTO        7960                                     : REM BRANCH TO EXIT
7820 REM **
7830  MAT COPY    Q2$()<769,192> TO B0$()<1,192>           : REM LOAD TEMP BUFFR
7840              B0$(5) = B0$(3)                          : REM LOAD TEMP BUFFR
7850              STR(B0$(2),30,34) = "the symbols, paying"
7860              STR(B0$(3),30,34) = "special attention to the"
7870              STR(B0$(4),30,34) = "ones you missed..."
7880  GOSUB ' 101 (7,1,10,0)                               : REM CLR CRT SCREEN
7890  $TRAN       (Q2$()<385,640>,  R4$()<177,4>)00 R      : REM CLR PAG BUFFER
7900  GOSUB ' 100 (07,1,64,B0$(1),0)                       : REM RE-LOAD TEXT
7910  GOSUB ' 100 (08,1,64,B0$(2),0)                       : REM RE-LOAD TEXT
7920  GOSUB ' 100 (09,1,64,B0$(3),0)                       : REM RE-LOAD TEXT
7930  GOSUB ' 100 (10,1,64,B0$(4),0)                       : REM RE-LOAD TEXT
7940  GOSUB ' 100 (16,1,64,B0$(5),0)                       : REM RE-LOAD TEXT
7950 REM **
7960  RETURN                                               : REM EXIT
```

```
7970 REM **
7980 REM **
7990 REM **  Subroutine  ********************************************** 2 1 4
8000 REM **  Abstract >
8010 REM **    To save lower part of page.
8020 REM **  Variables >
8030 REM **    BO$() > temp storage bufr ! Q2$() > page buffer
8040 REM **
8050 REM ***********************************************
8060  DEFFN ' 214                                          : REM ENTRY
8070  MAT COPY     Q2$()<961,64> TO BO$()<1,64>            : REM LOAD TEMP BUFR
8080  $TRAN        (Q2$()<961,64>, R4$()<177,4>)00 R       : REM CLR PAG BUFFR
8090  GOSUB ' 101 (16,1,1,0)                               : REM CLR CRT SCREEN
8100  RETURN                                               : REM EXIT
```

```
8110 REM **
8120 REM **
8130 REM **  Subroutine  ********************************************** 2 1 5
8140 REM **  Abstract >
8150 REM **    To write the name or rating of the group using this
8160 REM **    learning module.
8170 REM **  Variables >
8180 REM **    RO$ > begin sctr adrs ! RO > siz of text bufr
8190 REM **    R1$() > storage buffr ! Q2$() > pag buffer
8200 REM **    Q1 > row indx         ! Q2 > col indx
8210 REM **    R4$() > conversion buf!
8220 REM **
8230 REM ***********************************************
8240  DEFFN ' 215                                          : REM ENTRY
8250  GOSUB ' 105 (RO$,RO,00021)                           : REM FETCH TEXT
8260  IF          RO <= 160                  THEN 8290     : REM BUF SIZ OK
8270  $TRAN       (R1$()<160,1856>, R4$()<177,4>)00 R      : REM CLR STORAGE BUFR
8280              RO = 160                                 : REM SET TO MAXIUM
8290  MAT COPY    R1$()<1,RO+21> TO R1$()<RO+22,RO+21>     : REM POSITION BUFR TEXT
8300  MAT COPY    Q2$()<408,21> TO R1$()<1,RO+21>          : REM LOAD FIXED TEXT
8310  GOSUB ' 107 (07,24,60,RO$,0)                         : REM LOAD PAG BUFFER
8320  IF          Q2$(Q1,Q2-2) = "."         THEN 8340     : REM CHCK FOR PERIOD
8330  GOSUB ' 100 (Q1,Q2-1,1,".",0)                        : REM LOAD PERIOD
8340  RETURN                                               : REM EXIT
```

```
8350 REM **
8360 REM **
8370 REM **  Subroutine  ********************************************** 2 1 6
8380 REM **  Abstract >
8390 REM **    To save current Page.
8400 REM **  Variables >
8410 REM **    80$() > temp storage buffer !
8420 REM **
8430 REM ***********************************************
8440  DEFFN ' 216                                          : REM ENTRY
8450  GOSUB ' 100 (16,1,64,80$(1),0)                       : REM RE-LOAD TEXT
8460  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE PAGE
8470  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH308.1 - # MAP

6190 - 6440
6460 - 6390
6530 - 6170
7180 - 7110
7220 - 7100 7120
7510 - 7450
7830 - 7770
7960 - 7810
8290 - 8260
8340 - 8320

```
B0$(   - 7750 7830 7840 7850 7860 7870 7900 7910 7920 7930 7940 8070 8450
B1$(   - 6920 6980
B8     - 6980 6990 7030
D1     - 6370 7140 7470 8460
D2     - 6370 7140 7470 8460
D3     - 6130 6250 6330 6370 6420 6470 7140 7470 8460
D4$    - 6370 7140 7470 8460
D5     - 6370 7140 7470 8460
Q0(    - 6170 6390 6960 6980 7120 7400 7770
Q0$(   - 6930 7110 7170 7180 7200 7420 7510
Q1     - 7590 8320 8330
Q2     - 8320 8330
Q2$(   - 6670 7160 7490 7800 7830 7890 8070 8080 8300 8320
Q3$(   - 6990 7430 7510
Q4     - 6140 6260 6340 6430 6480
Q9$(   - 6990 7000 7010 7070
R0     - 7430 7440 7450 8250 8260 8280 8290 8300
R0$    - 6990 8250 8310
R1     - 7430
R1$(   - 8270 8290 8300
R2$(   - 6690 6700
R4$(   - 7050 7160 7490 7540 7800 7890 8080 8270
R5$(   - 6690 7000 7060 7550 7570
R9$(   - 6670 6680 6690 6710 6720 6730 6970 7040 7050 7060 7070 7080 7520 7530
         7540 7550 7560 7570 7580
T1     - 6150 6390 6710 7760 7770
T1$(   - 6910 7010 7020 7030 7410 7560
T2     - 6700 6710 6940 7080 7090 7100 7190 7390 7450 7500 7510 7580 7590
T3     - 6950 7080 7210
T4     - 7020 7070 7200 7210 7410 7430 7510
T5     - 6960 7010 7020 7030 7040 7120 7220 7400 7410 7530 7560 7600
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 30
       - 6530
6120 DEFFN' 50
???? DEFFN' 100
       - 6730 7080 7130 7460 7580 7900 7910 7920 7930 7940 8330 8450
???? DEFFN' 101
       - 7150 7480 7790 7880 8090
???? DEFFN' 102
       - 6110
???? DEFFN' 105
       - 7430 8250
???? DEFFN' 107
       - 7510 8310
???? DEFFN' 110
       - 6370 7140 7470 8460
???? DEFFN' 200
       - 6980
???? DEFFN' 205
       - 6240 6320 6410 6460
6660 DEFFN' 210
       - 6190 6270 6350
6900 DEFFN' 211
       - 6210
7380 DEFFN' 212
       - 6290
7740 DEFFN' 213
       - 6360
8060 DEFFN' 214
       - 6200 6280 6490
8240 DEFFN' 215
       - 6500
8440 DEFFN' 216
       - 6220 6300 6510
```

5.31 ATH3Ø9.Ø

The purpose of the 'ATH3Ø9.Ø' program is to create the Suggestions for Refresher Training pages of the learning module.

The remainder of this section presents the 'ATH3Ø9.Ø' program listing and reference maps.

```
6000 %## Program > AUTHOR/3   (ATH309.0)   Version > 2.0  2200 T/VP  780323
6010 REM **  Abstract  ************************************************************
6020 REM **                                                                        **
6030 REM **    This program will construct the frames describing                   **
6040 REM **    the suggestions for refresher training.                             **
6050 REM **                                                                        **
6060 REM ************************************************************************
6070 REM **
6080 REM **
6090  GOSUB ' 102 ("DATASUBPROGRAM","ATH109.0","9000"): REM LOAD DATA FORMATS
6100  DEFFN ' 050                                     : REM DEFINE ENTRY
6110              D3 = 32                             : REM CLR PAG INDX FLG
6120              Q4 = 46                             : REM SET ADRS INDX
6130  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
6140 REM **
6150  GOSUB ' 205 (1)                                 : REM LOAD NXT FORMAT
6160              D3 = 32                             : REM CLR PAG INDX FLG
6170              Q4 = 00                             : REM CLR ADRS INDX
6180  GOSUB ' 110 (D1,D2,D3,D4$,D5)                   : REM SAVE PAG ON DISK
6190 REM **
6200  GOSUB ' 205 (1)                                 : REM LOAD END OF FILE
6210              D3 = 32                             : REM CLR PAG INDX FLG
6220              Q4 = 49                             : REM SET ADRS INDX
6230  GOSUB ' 210                                     : REM SAVE END OF FILE
6240 REM **
6250  GOSUB ' 003                                     : REM RTURN TO MAIN PROG
```

```
6260 REM **
6270 REM **
6280 REM **  Subroutine  ********************************************** 2 1 0
6290 REM **  Abstract >
6300 REM **    To save last frame of learning module.
6310 REM **  Variables >
6320 REM **    D5 > output format index    ! Q4$() > pag adrs memory bufr
6330 REM **    QO() > numeric registr stk ! W1$ > binary page address
6340 REM **
6350 REM ***********************************************
6360  DEFFN ' 210                                          : REM ENTRY
6370  GOSUB ' 110 (D1,D2,D3,D4$,D5)                        : REM SAVE FRAME
6380              Q4$(47) = Q4$(49)                        : REM STOR LAST ADRS
6390  IF          D5 = 3                        THEN 6470  : REM SEQU INDX FORMAT
6400  GOSUB ' 103 (W1$,1,QO(6),QO(7)-(QO(8)-1),0,1)        : REM CMPUT FRST FCH ADR
6410              Q4$(47) = W1$                            : REM STOR FRST FCH ADRS
6420  GOSUB ' 103 (Q4$(49),W1,W2,W3,0,2)                   : REM CMPUT END FICH ADR
6430  GOSUB ' 103 (W1$,W1,QO(6),QO(7)-(QO(8)-1),0,1)       : REM CMPUT END FICH ADR
6440              Q4$(49) = W1$                            : REM STOR END FICH ADRS
6450              Q1$(D2) = W1$                            : REM INSRT INTO INDX
6460  DATA SAVE   BA T #3, (D1,R0)Q1$()                    : REM SAVE INDX BUFFER
6470  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH309.0 - # MAP

6470 - 6390

AUTHOR System: Phase 3 - ATH309.0 - V MAP

```
D1     - 6130 6180 6370 6460
D2     - 6130 6180 6370 6450
D3     - 6110 6130 6160 6180 6210 6370
D4$    - 6130 6180 6370
D5     - 6130 6180 6370 6390
Q0(    - 6400 6430
Q1$(   - 6450 6460
Q4     - 6120 6170 6220
Q4$(   - 6380 6410 6420 6440
R0     - 6460
W1     - 6420 6430
W1$    - 6400 6410 6430 6440 6450
W2     - 6420
W3     - 6420
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
       - 6250
6100 DEFFN' 50
???? DEFFN' 102
       - 6090
???? DEFFN' 103
       - 6400 6420 6430
???? DEFFN' 110
       - 6130 6180 6370
???? DEFFN' 205
       - 6150 6200
6360 DEFFN' 210
       - 6230
```

5.32 ATH3INDX

The purpose of the 'ATH3INDX' program is to update the disk lookup index on each output diskette if a multivolume output was created. The program will also scan each page of the learning module replacing all symbolic page references to the actual page number.

The 'ATH3INDX' program is presented in two parts.

5.32.1 ATH3IDX1

This part of the 'ATH3INDX' program contains the control logic for performing the functions described above. The 'ATH3IDX1' program also contains subroutines 201 to 213.

The remainder of this section presents the 'ATH3IDX1' program listing and reference maps.

AUTHOR System: Phase 3 - ATH31DX1 - CNTRL

```
4000 %## Program > AUTHOR/3  (ATH3IDX1)    Version > 2.0  2200 T/VP  780323
4004 REM **  Abstract  ************************************************************
4008 REM **                                                                        **
4012 REM **    This program will scan the phase III output file                    **
4016 REM **    and will replace all code references to there approprate            **
4020 REM **    frame address.  If a fiche format file was created,                 **
4024 REM **    then a complete index will be created for each fiche page           **
4028 REM **    created.  Before and after processing if more than one              **
4032 REM **    output disk was created then all index sectors on all disks         **
4036 REM **    will be updated.                                                    **
4040 REM **                                                                        **
4044 REM ************************************************************************
4048 REM **
4052 REM **
4056  GOSUB ' 102 ("DATASUBPROGRAM","ATH110.0","9000"): REM LOAD INDX FORMAT
4060  DEFFN ' 050                                     : REM DEFINE ENTRY
4064  DIM         T9$1                                : REM DEFINE EOF FLAG
4068              T9$ = HEX(FF)                       : REM INIT EOF FLAG
4072  GOSUB ' 201                                     : REM UPDATE DISK INDX
4076 REM **
4080  GOSUB ' 202                                     : REM LOAD PAGE FORMAT
4084  GOSUB ' 203                                     : REM TEST FOR INDEX
4088  GOSUB ' 204                                     : REM SCAN FOR KEYWORD
4092  GOSUB ' 205                                     : REM PROCESS KEYWORD
4096  GOSUB ' 206                                     : REM RE-SAVE PAGE
4100 REM **
4104   IF         T9$ = HEX(00)          THEN 4080     : REM LOAD NXT FORMAT
4108 REM **
4112              T9$ = HEX(FF)                       : REM SET LAST INDX FLG
4116  GOSUB ' 203                                     : REM INSRT LAST INDX
4120  GOSUB ' 201                                     : REM UPDATE DISK INDX
4124 REM **
4128              D1 = 474747                         : REM SET AUTO LIST FLG
4132  GOSUB ' 004                                     : REM RTURN TO MAIN PROG
```

```
4136 REM **
4140 REM **
4144 REM **  Subroutine  ********************************************* 2 0 1
4148 REM **  Abstract >
4152 REM **    To update all index sectors on all volumes of the
4156 REM **    output file, only if more than one volume was
4160 REM **    created.
4164 REM **  Variables >
4168 REM **    Q0$() > register stack          ! T1 > last volume id
4172 REM **    R0$() > input register          ! T2 > current volume indx
4176 REM **    Q9$() > i/o buffer              ! T3 > curretn volume id
4180 REM **    D0    > scratch indx            ! D1 > scratch indx
4184 REM **
4188 REM *********************************************
4192  DEFFN ' 201                                          : REM ENTRY
4196  MAT REDIM   Q9$(64)4                                 : REM DEFINE I/O BUFR
4200  IF          Q0$(2) <> HEX(31)          THEN 4432 : REM NO MULTI DISKS
4204              T2 = 1                                   : REM INIT VOL ID
4208  GOSUB ' 101 (1,1,16,0)                               : REM CLR CONSOLE SCREN
4212  GOSUB ' 101 (2,1,0,0)                                : REM POSITION CUSOR
4216  PRINTUSING  4400, " "                                : REM PRINT HEADING
4220  GOSUB ' 101 (6,1,0,0)                                : REM POSITION CUSOR
4224  PRINTUSING  4404, " "                                : REM PRINT INSTRUCTION
4228 REM **
4232  GOSUB ' 091                                          : REM RESPONCE WAIT
4236  DATA LOAD   BA T #3, (0,D0) Q9$()                    : REM LOAD SYS SECTOR
4240  ROTATE      (Q9$(13),4)                              : REM CONV TO DEC VAL
4244              T1 = VAL(STR(Q9$(13),1,1))               : REM STOR VOL ID
4248  IF          T1 > 1                         THEN 4280 : REM VALID VOL ID
4252  GOSUB ' 101 (10,1,0,0)                               : REM POSITION CUSOR
4256  PRINTUSING  4408, " "                                : REM PRINT ERR MSG
4260  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
4264  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
4268  GOSUB ' 101 (10,1,1,0)                               : REM ERASE ERR MSG
4272  GOTO        4232                                     : REM BRANCH TO WAIT
4276 REM **
4280  GOSUB ' 101 (6,1,1,0)                                : REM POSITION CUSOR
4284  PRINTUSING  4412, T2                                 : REM MOUNT DISK VOLUME
4288  GOSUB ' 091                                          : REM RESPONCE WAIT
4292  DATA LOAD   BA T #4, (0,D0) Q9$()                    : REM LOAD SYS SECTOR
4296  ROTATE      (Q9$(13),4)                              : REM CONV TO DEC VAL
4300              T3 = VAL(STR(Q9$(13),1,1))               : REM STOR VOL ID
4304  IF          T3 = T2                        THEN 4336 : REM VALID VOL ID
4308  GOSUB ' 101 (10,1,0,0)                               : REM POSITION CUSOR
4312  PRINTUSING  4416, T3, T2                             : REM PRINT ERR MSG
4316  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
4320  $GIO TONE   /005 (R8$(4),R8$(9))                     : REM PAUSE TONE
4324  GOSUB ' 101 (10,1,1,0)                               : REM ERASE ERR MSG
4328  GOTO        4288                                     : REM BRANCH TO WAIT
4332 REM **
4336  GOSUB ' 101 (6,1,1,0)                                : REM ERASE PROMPT
4340  PRINTUSING  4420, "00"                               : REM PRINT TITLE
```

```
4344 REM **
4348  FOR          DO = 1 TO 15                                  : REM BEGIN SAV LOOP
4352  DATA LOAD    BA T #3, (DO,D1) Q9$()                        : REM LOAD INDX SCTR
4356  DATA SAVE    BA T #4, (DO,D1) Q9$()                        : REM SAVE INDX SCTR
4360  GOSUB ' 101 (6,1,0,0)                                      : REM POSITION CUSOR
4364  PRINTUSING   4420, DO                                      : REM PRINT TITLE
4368  NEXT         DO                                            : REM CONT SAV LOOP
4372 REM **
4376               T2 = T2 + 1                                   : REM INCR VOL ID INDX
4380  IF           T2 = T1                          THEN 4432    : REM VOLUMES UPDATED
4384  GOTO         4280                                          : REM LOAD NXT DISK
4388 REM **
4392 REM **                                                      : REM IMAGE FORMATS
4396 REM **
4400 %  #   * *    U P D A T I N G    D I S K    I N D E X   * *
4404 %>>#   Mount Last Disk used into the PRIMARY DISK DRIVE        <<
4408 %>>#ERROR...LAST DISK VOLUME NOT MOUNTED OR NO MUTLIPLE DISKS <<
4412 %>>    Mount Volume ## Into the SECONDARY DISK DRIVE           <<
4416 %>>       ERROR...Volume ## is mounted ...  Mount Volume ##    <<
4420 %>>              Copying Index Sectors ...  ##                 <<
4424 REM **
4428 REM **
4432  RETURN                                                       : REM EXIT
```

```
4436 REM **
4440 REM **
4444 REM **  Subroutine  ********************************************** 2 0 2
4448 REM **  Abstract >
4452 REM **    To load page buffer with first/next frame from disk.
4456 REM **    If end of file set eof flag.
4460 REM **  Variables >
4464 REM **    D1 > indx sctr indx 1         ! R3$ > begin/next sctr addrs
4468 REM **    D2 > indx sctr indx 2         ! D0$ > scratch register
4472 REM **    L1 > indx sctr indx 1 (last) ! Q2$() > page buffer
4476 REM **    L2 > indx sctr indx 2 (last) ! T9$ > eof flag
4480 REM **    D0 > scratch register         ! Q1$() > indx buffer
4484 REM **
4488 REM ***********************************************
4492  DEFFN ' 202                                          : REM ENTRY
4496  IF          T9$ = HEX(00)                THEN 4520 : REM NOT FIRST TIME
4500              D1  = 1                                : REM INIT INDX SCTR
4504              D2  = -1                               : REM INIT SUB SCTR INDX
4508              L1  = 1                                : REM INIT INDX SCTR
4512              L2  = -3                               : REM INIT SUB SCTR INDX
4516  DATA LOAD   BA 1 #3, (D1,D0) Q1$()                 : REM LOAD INDX SCTR
4520              T9$ = HEX(00)                          : REM CLR EOF FLAG
4524              D2 = D2 + 2                            : REM INCR SUBSCTR INDX
4528              L2 = L2 + 2                            : REM INCR SUBSCTR INDX
4532  IF          L2 < 128                     THEN 4544 : REM NOT END OF SCTR
4536              L1 = L1 + 1                            : REM INCR SCTR ADDRS
4540              L2 = 1                                 : REM INIT SUBSCTR INDX
4544  IF          D2 < 128                     THEN 4584 : REM NOT END OF SCTR
4548              D1 = D1 + 1                            : REM INCR SCTR ADDRS
4552              D2 = 1                                 : REM INIT SUBSCTR INDX
4556  IF          D1 < 16                      THEN 4576 : REM VALID INDX SCTR
4560              T9$ = HEX(EF)                          : REM SET EOF FLAG
4564              D1 = D1 - 1                            : REM RESET TO LAST SCTR
4568  GOTO        4656                                   : REM BRANCH TO EXIT
4572 REM **
4576  DATA LOAD   BA T #3, (D1,D0) Q1$()                 : REM LOAD INDX SCTR
4580 REM **
4584  IF          Q1$(D2) <> Q4$(49)           THEN 4596 : REM NOT END OF FILE
4588              T9$ = HEX(EF)                          : REM SET EOF FLAG
4592  GOTO        4656                                   : REM BRANCH TO EXIT
4596  GOSUB ' 106 (Q1$(D2+1),2)                          : REM MOUNT DISK
4600 REM **
4604  FOR         D0 = 1 TO 4                            : REM BEGIN INPUT LOOP
4608  DATA LOAD   BA T #3, (R3$,R3$) Q9$()               : REM LOAD PAGE PART
4612  MAT COPY    Q9$() TO Q2$()<(D0-1)*256+1,256>       : REM LOAD INTO PAG BUFR
4616  NEXT        D0                                     : REM CONT INPUT LOOP
4620 REM **
4624  IF          T9$ = HEX(EF)                THEN 4656 : REM END OF FILE
4628  PRINT       HEX(03)                                : REM CLR CRT SCREEN
4632              R8$ = HEX(5102)                        : REM SET PACK FORMAT
4636  FOR         R9 = 1 TO 16                           : REM BEG BUFR PRINT
4640  $PACK       (F=R8$) STR(R8$(9),7,2) FROM R9        : REM PACK LINE INDX
4644  $CIO /005   (R8$(6),R8$(9))Q2$()<(R9-1)*64+1,64>: REM PRINT BUFFER
4648  NEXT        R9                                     : REM CONT BUFR PRINT
4652 REM **
4656  RETURN                                             : REM EXIT
```

```
4660 REM **
4664 REM **
4668 REM **  Subroutine  ********************************************** 2 0 3
4672 REM **  Abstract >
4676 REM **    To check current page address and to create the
4680 REM **    index pages if fiche format output.
4684 REM **  Variables >
4688 REM **    T9$ > eof flag                ! W3 > fiche col number
4692 REM **    Q1$() > index buffer          ! Q0() > numeric register stack
4696 REM **    D2 > indx bufr indx 2         ! Q4$() > page ref memory bufr
4700 REM **    W1 > fiche page number        ! Q8$() > heading memory bufr
4704 REM **    W2 > fiche row number         ! D5 > file format indx
4708 REM **
4712 REM ***********************************************
4716  DEFFN ' 203                                        : REM ENTRY
4720  DIM          Q8$(4)64                              : REM DEFINE WORK BUFRS
4724  INIT(SF)     Q8$()                                 : REM INIT HEADING TBL
4728  Q8$(1) =     "Practicing All Symbols"              : REM DEFINE TABLE
4732  Q8$(2) =     "Criterion Tests"                     : REM DEFINE TABLE
4736  Q8$(3) =     "Suggestions For Refresher Training": REM DEFINE TABLE
4740 REM **
4744  IF           T9$ = HEX(EF)            THEN 4816 : REM END OF FILE
4748  GOSUB ' 103 (Q1$(D2),W1,W2,W3,0,2)             : REM CONV PAGE ADDRESS
4752  IF           T9$ = HEX(FF)            THEN 4772 : REM FINAL INDEX
4756  IF           D5 = 3                   THEN 4816 : REM LINEAR FORMAT FILE
4760  IF           W2 <> Q0(6)              THEN 4816 : REM NOT LAST ROW
4764  IF           W3 <> Q0(7)-(Q0(8)-1)    THEN 4816 : REM NOT LAST COL
4768 REM **
4772  GOSUB ' 208                                    : REM UPDATE INDX
4776  GOSUB ' 209 (1)                                : REM LOAD INDX FORMAT
4780  GOSUB ' 210 (Q4$(41),3,57)                     : REM INSRT INTRODUCTION
4784  GOSUB ' 210 (Q4$(42),5,57)                     : REM INSRT OPT TEST
4788  GOSUB ' 210 (Q4$(43),7,57)                     : REM INSRT ORG OF PROG
4792  GOSUB ' 211                                    : REM INSRT SBL SETS
4796  GOSUB ' 212 (1,44)                             : REM INSRT PRAC SBLS
4800  GOSUB ' 212 (2,45)                             : REM INSRT CRITRON TST
4804  GOSUB ' 212 (3,46)                             : REM INSRT SUGGESTIONS
4808  GOSUB ' 213                                    : REM INSRT END OF INDX
4812 REM **
4816  RETURN                                         : REM EXIT
```

```
4820 REM **
4824 REM **
4828 REM **  Subroutine  ********************************************** 2 0 4
4832 REM **  Abstract >
4836 REM **    To scan the current page buffer for a keyword. If
4840 REM **    a keyword in found set up key buffer. Process all
4844 REM **    keywords on page.
4848 REM **  Variables >
4852 REM **    Q5$() > keyword indx buffer  ! R5$() > single char buffer
4856 REM **    T8$() > keyword info buffer  ! R2$() > srch return buffer
4860 REM **    T8 > keyword bufr indx       ! R2 > position in index
4864 REM **    T7 > page buffer indx        ! T9$ > eof flag
4868 REM **
4872 REM ***********************************************
4876  DEFFN ' 204                                          : REM ENTRY
4880  DIM          T8$(10,3)2                              : REM DEFIN WORK BUFRS
4884  DEFFN        P(P) = INT(P/256)                       : REM DEFINE FUNCTION
4888  DEFFN        Q(Q) = Q - FNP(Q)*256                   : REM DEFINE FUNCTION
4892 REM **
4896  IF           T9$ = HEX(EF)                THEN 5192  : REM END OF FILE
4900  INIT(00)     T8$()                                   : REM INIT KEY BUFR
4904               T8 = 1                                  : REM INIT KEY INDX
4908               T7 = 1                                  : REM INIT BUFR INDX
4912  IF           T7 > 1024                    THEN 4992  : REM END OF PAGE
4916  MAT SEARCH   Q2$()<T7,1024-T7+1>,=R5$(5) TO R2$()    : REM SRCH FOR KEYWORD
4920  IF           R2$(1) = HEX(0000)           THEN 4992  : REM NO KEYWRDS FOUND
4924               T7 = FNR(VAL(R2$(1))) + T7              : REM INCR BUFR INDX
4928  MAT SEARCH   Q2$()<T7,1024-T7+1>,=R5$(1) TO R2$()    : REM SRCH FOR END
4932               T6 = FNR(VAL(R2$(1))) - 1               : REM LENGTH OF KEY
4936  IF           T6 <> 4                      THEN 4984  : REM INVALID KEY LEN
4940  MAT COPY     Q2$()<T7,T6> TO R9$()<1,64>             : REM STOR KEYWORD
4944  MAT SEARCH   Q5$(), = STR(R9$(1),1,3) TO R2$()       : REM SRCH KEYWRD BUFR
4948  IF           R2$(1) = HEX(0000)           THEN 4984  : REM INVALID KEY
4952               R2 = FNR(VAL(R2$(1)))                   : REM CMPUT BUFR POS
4956               R2 = INT(R2/3) + 1                      : REM CMPUT INDX
4960               BIN(STR(T8$(T8,1),1,1)) = FNP(T7-1)     : REM STOR BUFR POSITION
4964               BIN(STR(T8$(T8,1),2,1)) = FNQ(T7-1)     : REM STOR BUFR POSITION
4968               BIN(STR(T8$(T8,2),1,1)) = FNP(R2)       : REM STOR INDEX
4972               BIN(STR(T8$(T8,2),2,1)) = FNQ(R2)       : REM STOR INDEX
4976               STR(T8$(T8,3),1,2) = STR(R9$(1),4,1)    : REM STOR KEYWORD PRAM
4980               T8 = T8 + 1                             : REM INCR KEY BUFR INDX
4984               T7 = T7 + T6                            : REM INCR BUFR INDX
4988  GOTO         4912                                    : REM SRCH NXT KEYWORD
4992  IF           T8 = 1                       THEN 5004  : REM ONLY ONE KEYWORD
4996               T8 = T8 + 1                             : REM ADJUST INDEX
5000 REM **
5004  RETURN                                               : REM EXIT
```

```
5008 REM **
5012 REM **
5016 REM **  Subroutine  ********************************************** 2 0 5
5020 REM **  Abstract >
5024 REM **    To insert frame address corresponding to the keyword.
5028 REM **  Variables >
5032 REM **    T8$() > keyword data buffer    ! T8 > keyword bufr indx
5036 REM **    T9$   > eof flag               ! T6 > branch indx
5040 REM **    T8$   > binary page address    ! S1 > page row location
5044 REM **    S2    > page col location      !
5048 REM **
5052 REM ***********************************************
5056  DEFFN ' 205                                          : REM ENTRY
5060  DIM          T8$2                                    : REM DEFINE WORK BUFRS
5064  DEFFN        S(S) = 256*VAL(STR(T8$(S,1),1,1))+FNT(S)  : REM FUNCTION
5068  DEFFN        T(T) =     VAL(STR(T8$(T,1),2,1))         : REM FUNCTION
5072  DEFFN        U(U) = 256*VAL(STR(T8$(U,2),1,1))+FNV(U)  : REM FUNCTION
5076  DEFFN        V(V) =     VAL(STR(T8$(V,2),2,1))         : REM FUNCTION
5080 REM **
5084               T8   = 1                                : REM INIT KEY INDX
5088  IF           T9$  = HEX(EF)            THEN 5192     : REM END OF FILE
5092  IF           T8$(T8,1) = HEX(0000)     THEN 5192     : REM END OF KEYS
5096               T6   = FNU(T8)                          : REM FETCH KEY INDX
5100 REM **
5104 REM **        ! skip b ! skip f ! index  ! close  ! practc !
5108 REM **        !--------!--------!--------!--------!--------!
5112  ON T6 GOTO     5128  ,  5132  ,  5136  ,  5140  ,  5144  ,
                     5148  ,  5152  ,  5156  ,  5160  ,  5164
5116 REM **        !--------!--------!--------!--------!--------!
5120 REM **        !   end  !  diag  !  extra !  extra !  extra !
5124 REM **
5128  GOSUB ' 214 (1,T8,T8$)              : GOTO 5172 : REM SKIP BACK
5132  GOSUB ' 214 (2,T8,T8$)              : GOTO 5172 : REM SKIP FORWARD
5136  GOSUB ' 215 (1,T8$)                 : GOTO 5172 : REM INSRT INDX
5140  GOSUB ' 215 (2,T8$)                 : GOTO 5172 : REM INSRT CLOSE
5144  GOSUB ' 215 (3,T8$)                 : GOTO 5172 : REM INSRT PRACTICE
5148  GOSUB ' 215 (4,T8$)                 : GOTO 5172 : REM INSRT END
5152  GOSUB ' 215 (5,T8$)                 : GOTO 5184 : REM DIAGRAM
5156  GOSUB ' 215 (6,T8$)                 : GOTO 5184 : REM EXTRA SLOT
5160  GOSUB ' 215 (7,T8$)                 : GOTO 5184 : REM EXTRA SLOT
5164  GOSUB ' 215 (8,T8$)                 : GOTO 5184 : REM EXTRA SLOT
5168 REM **
5172               S1 = INT((FNS(T8)-1)/64) + 1          : REM CMPUTE ROW INDX
5176               S2 = FNS(T8) - (S1-1)*64              : REM CMPUTE COL INDX
5180  GOSUB ' 210 (T8$,S1,S2)                            : REM INSRT PAG ADRS
5184               T8 = T8 + 1                           : REM INCR KEY INDX
5188  GOTO         5092                                  : REM GET NEXT KEY
5192  RETURN                                             : REM EXIT
```

```
5196 REM **
5200 REM **
5204 REM **  Subroutine  ********************************************** 2 0 6
5208 REM **  Abstract >
5212 REM **    To save current page on disk if any changes were made.
5216 REM **  Variables >
5220 REM **    T9$ > eof flag              ! T8$() > keyword data buffer
5224 REM **    Q1$() > disk indx buffer    ! D2 > indx bufr indx 2
5228 REM **    Q2$() > page buffer         ! D0 > loop indx
5232 REM **    Q9$() > i/o buffer          ! R3$ > disk storage sctr addrs
5236 REM **
5240 REM ***********************************************
5244  DEFFN ' 206                                          : REM ENTRY
5248  IF          T9$ = HEX(EF)              THEN 5284     : REM END OF FILE
5252  IF          T8$(1,1) = HEX(0000)       THEN 5284     : REM NO CHANGES
5256  GOSUB ' 106 (Q1$(D2+1),2)                            : REM MOUNT DISK
5260 REM **
5264  FOR         D0 = 1 TO 4                              : REM BEG STOR LOOP
5268  MAT COPY    Q2$()<(D0-1)*256+1,256> TO Q9$()         : REM LOAD I/O BUFR
5272  DATA SAVE   BA T #3, (R3$,R3$) Q9$()                 : REM SAVE BUFFER
5276  NEXT        D0                                       : REM CONT STOR LOOP
5280 REM **
5284  RETURN                                               : REM EXIT
```

```
5288 REM **
5292 REM **
5296 REM **  Subroutine  ********************************************** 2 0 8
5300 REM **  Abstract >
5304 REM **    To update disk indx and indx pointers for spaces
5308 REM **    occupied by the fiche indx frames.
5312 REM **  Variables >
5316 REM **    L3 > indx sctr addrs (temp) ! DO > loop indx
5320 REM **    L4 > indx sctr addrs (temp) ! Q1$() > disk indx buffer
5324 REM **    L5 > scratch sctr addrs     ! L5$ > beging page address
5328 REM **
5332 REM ************************************************
5336  DEFFN ' 208                                          : REM ENTRY
5340  DIM          L5$2                                    : REM DEFINE WORK BUFRS
5344               L3 = D1                                 : REM INIT INDX ADDRS
5348               L4 = D2                                 : REM INIT BUFR INDX
5352               L5$ = Q1$(D2)                           : REM INIT BEG PAG ADRS
5356 REM **
5360  FOR          DO = 1 TO Q0(8)                         : REM BEG PAG LOOP
5364  GOSUB ' 103 (L5$,W1,W2,W3,(DO-1),3)                  : REM CREATE PAG ADRS
5368               Q1$(L4) = W1$                           : REM STOR PAG ADRS
5372  DATA SAVE    BA T #3, (L3,L5) Q1$()                  : REM SAVE INDX SCTR
5376               L4 = L4 + 2                             : REM INCR BUFR INDX
5380  IF           L4 < 128                  THEN 5396     : REM BUFR NOT FULL
5384               L3 = L3 + 1                             : REM INCR SCRT ADDRS
5388               L4 = 1                                  : REM RESET BUFR INDX
5392  DATA LOAD    BA T #3, (L3,L5) Q1$()                  : REM LOAD INDX SCTR
5396  NEXT         DO                                      : REM CONT PAG LOOP
5400 REM **
5404  DATA LOAD    BA T #3, (D1,DO) Q1$()                  : REM RELOAD CURNT SCTR
5408  RETURN                                               : REM EXIT
```

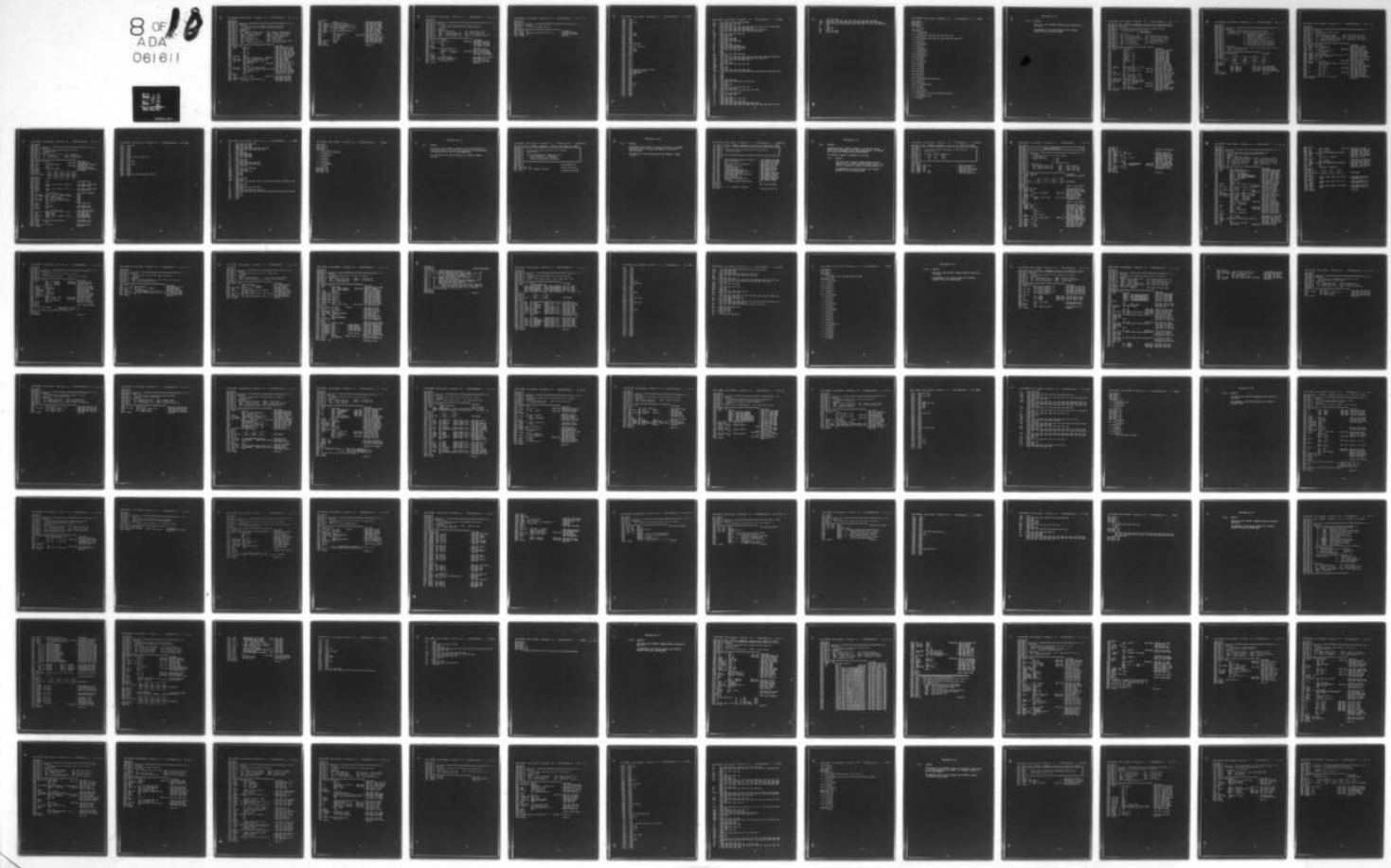
```
5412 REM **
5416 REM **
5420 REM **  Subroutine  ********************************************** 2 0 9
5424 REM **  Abstract >
5428 REM **    To Initialize output page buffer from the
5432 REM **    Data Subprogram.
5436 REM **  Variables >
5440 REM **    CO > initial buf indx  ! Q2$() > page buffer
5444 REM **    QO() > prog registr stk!
5448 REM **
5452 REM ***********************************************
5456  DEFFN ' 209 (CO)                                    : REM ENTRY
5460  RESTORE     1                                       : REM RESTOR POINTER
5464 REM **
5468 REM **       ! CLR BUFR  !  LOAD BEG  !  SKP FMT : REM FUNCTION 1
5472 REM **       ! LOAD BEG  !  PRINT TXT !  NULL    : REM FUNCTION 2
5476 REM **       ! PRINT TXT !  NULL      !  NULL    : REM FUNCTION 3
5480 REM **       !-----------!------------!----------!
5484  ON CO GOTO      5500     ,   5504     ,   5516   : REM BRANCH
5488 REM **       !-----------!------------!----------!
5492 REM **
5496 REM **
5500  INIT        (20) Q2$()                            : REM INITIAL BUFR
5504              QO(9) = 1                             : REM SET TO BUFR BEG
5508              QO(10) = 1024                         : REM SET BUFR SIZE
5512  $GIO        ERASE CRT/005(4003,R8$(9))            : REM CLEAR CRT SCRN
5516  GOSUB ' 217 (CO)                                  : REM LOAD BUFR
5520  RETURN                                            : REM EXIT
```

```
5524 REM **
5528 REM **
5532 REM **  Subroutine  ********************************************** 2 1 0
5536 REM **  Abstract >
5540 REM **    To convert the binary page address into proper
5544 REM **    format and insert into page buffer at given location.
5548 REM **  Variables >
5552 REM **    S1 > page row location       ! S2 > page col location
5556 REM **    D8$ > binary page address    ! D7$ > converted page address
5560 REM **
5564 REM ************************************************
5568  DEFFN ' 210 (D8$,S1,S2)                                  : REM ENTRY
5572  IF          D8$ <> HEX(FFFF)           THEN 5588 : REM ITM INDX EXISTS
5576              D7$ =  " N/A"                                : REM SET PAGE REFERNCE
5580  GOTO        5592                                         : REM INSRT PAGE REFRNCE
5584 REM **
5588  GOSUB ' 118 (D8$,D7$)                                    : REM CONV PAG ADDRS
5592  GOSUB ' 100 (S1,S2,5,D7$,0)                              : REM INSRT INTO BUFR
5596  RETURN                                                   : REM EXIT
```

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U)
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

8 OF 10
ADA
061611



```
5600 REM **
5604 REM **
5608 REM **  Subroutine  ********************************************** 2 1 1
5612 REM **  Abstract >
5616 REM **    To create the list of symbols by symbol set to use
5620 REM **    in the fiche index and insert these lists into the
5624 REM **    page buffer.
5628 REM **  Variables >
5632 REM **    M1 > symbol id table indx 1  ! M2 > symbol id table indx 2
5636 REM **    Q1 > page bufer line indx    ! M3 > internal line position
5640 REM **    Q7$() > symbol id table      ! Q3$() > symbol lokup table
5644 REM **    Q9$() > i/o buffer           ! R9$() > temp line buffer
5648 REM **    D0 > loop index              ! M4 > symbol length
5652 REM **    R1$() > storage bufer        ! R0 > buffer size
5656 REM **    R1 > item id                 ! Q0$() > system register stack
5660 REM **    B0$() > temp storage buffer  ! B0 > storage bufr indx
5664 REM **
5668 REM ************************************************
5672  DEFFN ' 211                                        : REM ENTRY
5676  DIM          B0$(6)45                              : REM DEFINE WORK BUFRS
5680               M1 = 1                                : REM INIT SBL ID INDX
5684               M2 = 1                                : REM INIT SBL ID INDX
5688               M3 = 1                                : REM INIT BUFR POS INDX
5692               Q1 = 10                               : REM INIT BUFR LIN INDX
5696               B0 = 1                                : REM INIT STOR BUFR IDX
5700  INIT(20)     B0$()                                 : REM CLR STORAGE BUFR
5704  IF           Q7$(M2,1) <> Q7$(M1,1)    THEN 5756   : REM END OF SBL SET
5708  DATA LOAD    BA T #2, (Q3$(M2,3),R0$) Q9$()        : REM LOAD SYMBOL
5712  MAT COPY     Q9$()<1,15> TO R9$()<1,64>            : REM STOR SYMBOL
5716               M4 = LEN(R9$(1))                      : REM CMPUTE SBL LENGTH
5720  IF           M3+M4+3 < 41              THEN 5732   : REM LINE NOT FULL
5724               M3 = 1                                : REM RESET BUFR POS IDX
5728               B0 = B0 + 1                           : REM INCR BUFR INDX
5732  MAT COPY     R5$()<9,1> TO R9$()<M4+2,1>           : REM STOR SEP CHAR
5736  MAT COPY     R9$()<1,M4+2> TO B0$()<(B0-1)*45+M3,M4+2>: REM STOR SYMBOL
5740               M3 = M3 + M4 + 3                      : REM INCR LIN POSITION
5744               M2 = M2 + 1                           : REM INCR SBL TBL INDX
5748  GOTO         5704                                  : REM GET NEXT SYMBOL
5752 REM **
5756  IF           16-Q1 >= B0+1             THEN 5776   : REM SPACE AVAIABLE
5760  GOSUB ' 216 (1)                                    : REM SAVE INDX PAGE
5764  GOSUB ' 100 (3,6,13,"Symbol Sets:",0)              : REM INSRT HEADING
5768               Q1 = 4                                : REM RESET LINE INDX
```

```
5772 REM **
5776              Q0$(1) = "1"                                  : REM SET NUL LOOKUP
5780  GOSUB ' 105 (Q3$(M1,2),R0,R1)                             : REM FETCH SET NAME
5784  MAT COPY    R1$()<1,64> TO R9$()<1,64>                    : REM STOR SET NAME
5788  GOSUB ' 100 (Q1,10,40,R9$(1),0)                           : REM INSRT SET NAME
5792  GOSUB ' 210 (Q4$(VAL(Q7$(M1,1))),Q1,57)                   : REM INSRT PAG ADDRS
5796  FOR         D0 = 1 TO B0                                  : REM BEG SBL INSRT
5800  GOSUB ' 100 (Q1+D0,15,40,B0$(D0),0)                       : REM INSRT SYMBOLS
5804  NEXT        D0                                            : REM CONT SBL INSRT
5808              Q1 = Q1 + B0 + 1                              : REM INCR LINE INDX
5812  IF          M2 > Q0(2)                  THEN 5840         : REM END OF SETS
5816              M1 = M2                                       : REM SET SBL TBL INDX
5820              M3 = 01                                       : REM RESET LINE POS
5824              B0 = 01                                       : REM RESET BUFR INDX
5828  INIT(20)    B0$()                                         : REM CLL STOR BUFFR
5832  GOTO        5704                                          : REM PROCSS NXT SET
5836 REM **
5840  RETURN                                                    : REM EXIT
```

```
5844 REM **
5848 REM **
5852 REM **  Subroutine  ********************************************** 2 1 2
5856 REM **  Abstract >
5860 REM **    To insert an index item into the index if there
5864 REM **    is enough space, otherwise skip to next page then
5868 REM **    insert.
5872 REM **  Variables >
5876 REM **    R9$() > text buffer           ! Q8 > heading bufr index
5880 REM **    D7$   > converted pag adrs    ! Q1 > buffer line indx
5884 REM **    Q8$() > heading memory bufr   ! Q4 > pag reference bufr indx
5888 REM **    Q4$() > pag ref memory bufr   ! Q0$() > sys memory flags
5892 REM **
5896 REM ***********************************************
5900  DEFFN ' 212 (Q8,Q4)                                  : REM ENTRY
5904  IF          16-Q1 >= 2                THEN 5924    : REM SPACE AVAIABLE
5908  GOSUB ' 216 (1)                                      : REM SAVE INDX PAGE
5912              Q1 = 3                                   : REM RESET LINE INDX
5916  GOTO        5932                                     : REM BRANCH TO INSRT
5920 REM **
5924  GOSUB ' 100 (Q1,6,54,Q8$(4),0)                       : REM INSRT LINE
5928              Q1 = Q1 + 1                              : REM INCR LINE INDX
5932  IF          Q4$(Q4) <> HEX(FFFF)      THEN 5948    : REM ITM INDX EXISTS
5936              D7$       = " N/A"                       : REM SET PAG REFERNCE
5940  GOTO        5952                                     : REM INSRT PAG REFRNCE
5944 REM **
5948  GOSUB ' 118 (Q4$(Q4),D7$)                            : REM CREATE PAG REFRNCE
5952  GOSUB ' 100 (Q1,6,40,Q8$(Q8),0)                      : REM INSRT TEXT
5956  GOSUB ' 100 (Q1,57,4,D7$,0)                          : REM INSRT PAG ADRS
5960              Q1 = Q1 + 1                              : REM INCR LINE INDX
5964  RETURN                                               : REM EXIT
```

```
5968 REM **
5972 REM **
5976 REM **  Subroutine  ********************************************** 2 1 3
5980 REM **  Abstract >
5984 REM **    To save last page of index.
5988 REM **
5992 REM ********************************************
5996  DEFFN ' 213                                          : REM ENTRY
6000  GOSUB ' 100 (16,46,18,"End of Index",0)              : REM INSRT MESSAGE
6004  GOSUB ' 216 (0)                                      : REM SAVE INDX PAGE
6008  RETURN                                               : REM EXIT
```

```
4080  - 4104
4232  - 4272
4280  - 4248 4384
4288  - 4328
4336  - 4304
4400  - 4216
4404  - 4224
4408  - 4256
4412  - 4284
4416  - 4312
4420  - 4340 4364
4432  - 4200 4380
4520  - 4496
4544  - 4532
4576  - 4556
4584  - 4544
4596  - 4584
4656  - 4568 4592 4624
4772  - 4752
4816  - 4744 4756 4760 4764
4912  - 4988
4984  - 4936 4948
4992  - 4912 4920
5004  - 4992
5092  - 5188
5128  - 5112
5132  - 5112
5136  - 5112
5140  - 5112
5144  - 5112
5148  - 5112
5152  - 5112
5156  - 5112
5160  - 5112
5164  - 5112
5172  - 5128 5132 5136 5140 5144 5148
5184  - 5152 5156 5160 5164
5192  - 4896 5088 5092
5284  - 5248 5252
5396  - 5380
5500  - 5484
5504  - 5484
5516  - 5484
5588  - 5572
5592  - 5580
5704  - 5748 5832
5732  - 5720
5756  - 5704
5776  - 5756
5840  - 5812
5924  - 5904
5932  - 5916
5948  - 5932
5952  - 5940
```

```
B0     - 5696 5728 5736 5756 5796 5808 5824
B0$(   - 5676 5700 5736 5800 5828
C0     - 5456 5484 5516
D0     - 4236 4292 4348 4352 4356 4364 4368 4516 4576 4604 4612 4616 5264 5268
         5276 5360 5364 5396 5404 5796 5800 5804
D1     - 4128 4352 4356 4500 4516 4548 4556 4564 4576 5344 5404
D2     - 4504 4524 4544 4552 4584 4596 4748 5256 5348 5352
D5     - 4756
D7$    - 5576 5588 5592 5936 5948 5956
D8$    - 5568 5572 5588
L1     - 4508 4536
L2     - 4512 4528 4532 4540
L3     - 5344 5372 5384 5392
L4     - 5348 5368 5376 5380 5388
L5     - 5372 5392
L5$    - 5340 5352 5364
M1     - 5680 5704 5780 5792 5816
M2     - 5684 5704 5708 5744 5812 5816
M3     - 5688 5720 5724 5736 5740 5820
M4     - 5716 5720 5732 5736 5740
P      - 4884
Q      - 4888
Q0(    - 4760 4764 5360 5504 5508 5812
Q0$(   - 4200 5776
Q1     - 5692 5756 5768 5788 5792 5800 5808 5904 5912 5924 5928 5952 5956 5960
Q1$(   - 4516 4576 4584 4596 4748 5256 5352 5368 5372 5392 5404
Q2$(   - 4612 4644 4916 4928 4940 5268 5500
Q3$(   - 5708 5780
Q4     - 5900 5932 5948
Q4$(   - 4584 4780 4784 4788 5792 5932 5948
Q5$(   - 4944
Q7$(   - 5704 5792
Q8     - 5900 5952
Q8$(   - 4720 4724 4728 4732 4736 5924 5952
Q9$(   - 4196 4236 4240 4244 4292 4296 4300 4352 4356 4608 4612 5268 5272 5708
         5712
R0     - 5780
R0$    - 5708
R1     - 5780
R1$(   - 5784
R2     - 4952 4956 4968 4972
R2$(   - 4916 4920 4924 4928 4932 4944 4948 4952
R3$    - 4608 5272
R5$(   - 4916 4928 5732
R8$    - 4632 4640
R8$(   - 4260 4264 4316 4320 4640 4644 5512
R9     - 4636 4640 4644 4648
R9$(   - 4940 4944 4976 5712 5716 5732 5736 5784 5788
S      - 5064
S1     - 5172 5176 5180 5568 5592
S2     - 5176 5180 5568 5592
T      - 5068
T1     - 4244 4248 4380
T2     - 4204 4284 4304 4312 4376 4380
T3     - 4300 4304 4312
T6     - 4932 4936 4940 4984 5096 5112
T7     - 4908 4912 4916 4924 4928 4940 4960 4964 4984
T8     - 4904 4960 4964 4968 4972 4976 4980 4992 4996 5084 5092 5096 5128 5132
```

```
          5172 5176 5184
T8$     - 5060 5128 5132 5136 5140 5144 5148 5152 5156 5160 5164 5180
T8$(    - 4880 4900 4960 4964 4968 4972 4976 5064 5068 5072 5076 5092 5252
T9$     - 4064 4068 4104 4112 4496 4520 4560 4588 4624 4744 4752 4896 5088 5248
U       - 5072
V       - 5076
W1      - 4748 5364
W1$     - 5368
W2      - 4748 4760 5364
W3      - 4748 4764 5364
```

AUTHOR System: Phase 3 - ATH31DX1 - ' MAP

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 4
       - 4132
4060 DEFFN' 50
???? DEFFN' 91
       - 4232 4288
???? DEFFN' 100
       - 5592 5764 5788 5800 5924 5952 5956 6000
???? DEFFN' 101
       - 4208 4212 4220 4252 4268 4280 4308 4324 4336 4360
???? DEFFN' 102
       - 4056
???? DEFFN' 103
       - 4748 5364
???? DEFFN' 105
       - 5780
???? DEFFN' 106
       - 4596 5256
???? DEFFN' 118
       - 5588 5948
4192 DEFFN' 201
       - 4072 4120
4492 DEFFN' 202
       - 4080
4716 DEFFN' 203
       - 4084 4116
4876 DEFFN' 204
       - 4088
5056 DEFFN' 205
       - 4092
5244 DEFFN' 206
       - 4096
5336 DEFFN' 208
       - 4772
5456 DEFFN' 209
       - 4776
5568 DEFFN' 210
       - 4780 4784 4788 5180 5792
5672 DEFFN' 211
       - 4792
5900 DEFFN' 212
       - 4796 4800 4804
5996 DEFFN' 213
       - 4808
???? DEFFN' 214
       - 5128 5132
???? DEFFN' 215
       - 5136 5140 5144 5148 5152 5156 5160 5164
???? DEFFN' 216
       - 5760 5908 6004
???? DEFFN' 217
       - 5516
```

5.32.2 ATH3IDX2

This part of the 'ATH3INDX' program contains subroutines 214 to 217.

The remainder of this section presents the 'ATH3IDX2' program listing and reference maps.

```
6400 %## Program > AUTHOR/3   (ATH3IDX2)   Version > 2.0  2200 T/VP  780323
6404 REM **  Subroutine  ********************************************** 2 1 4
6408 REM **  Abstract >
6412 REM **    To perform the skip foward or skip backward function
6416 REM **    depending on function key index.
6420 REM **          Key index      1 - skip backward
6424 REM **                         2 - skip forward
6428 REM **  Variables >
6432 REM **    T7 > function key index     ! T8 > keyword buffer index
6436 REM **    T8$ > binary page adrs      ! T8() > function index limits
6440 REM **    M1 > indx sctr adrs (temp)  ! M2 > indx bufr indx (temp)
6444 REM **    T6 > sections to skip       ! T5 > sections skipped
6448 REM **    Q9$() > i/o buffer          ! D0 > scratch adrs register
6452 REM **    D1 > indx sctr adrs         ! D2 > indx bufr indx
6456 REM **    Q1$() > disk indx buffr     !
6460 REM **
6464 REM ***********************************************
6468  DEFFN ' 214 (T7,T8,T8$)                                  : REM ENTRY
6472  DIM         T8(2,4)                                      : REM DEFINE WORK BUFR
6476              T8(1,1) = -1                                 : REM DEFINE TABLE
6480              T8(1,2) =  1                                 : REM DEFINE TABLE
6484              T8(1,3) =  0                                 : REM DEFINE TABLE
6488              T8(1,4) =  129                               : REM DEFINE TABLE
6492              T8(2,1) =  1                                 : REM DEFINE TABLE
6496              T8(2,2) =  127                               : REM DEFINE TABLE
6500              T8(2,3) =  16                                : REM DEFINE TABLE
6504              T8(2,4) = -1                                 : REM DEFINE TABLE
6508              T5      =  0                                 : REM INIT COUNTER
6512              M1      =  D1                                : REM INIT SCTR ADRS
6516              M2      =  D2                                : REM INIT BUFR INDX
6520  CONVERT     STR(T8$(T8,3),1,1) TO T6                     : REM SECTIONS TO SKP
6524  IF          M2 <> T8(T7,2)                THEN 6548      : REM NOT END OF BUFFR
6528              M1 =  M1 + T8(T7,1)*1                        : REM INCR SCTR ADRS
6532  IF          M1 <> T8(T7,3)                THEN 6540      : REM NOT END OF INDX
6536  GOTO        6576                                         : REM INDX FLG NOT FOUND
6540  DATA LOAD   BA T #3, (M1,D0) Q1$()                       : REM LOAD INDX SCTR
6544              M2 =  T8(T7,4)                               : REM RESET BUFR INDX
6548              M2 =  M2 + T8(T7,1)*2                        : REM INCR BUFR INDX
6552  GOSUB ' 106 (Q1$(M2+1),2)                                : REM LOAD DISK VOL
6556  DATA LOAD   BA T #3, (R3$,R3$) Q9$()                     : REM LOAD TOP OF PAGE
6560  MAT COPY    Q9$()<63,1> TO R0$()<1,1>                    : REM FETCH PAG INDX FLG
6564  IF          R0$(1) <> HEX(00)             THEN 6524      : REM INDX FLG NOT SET
6568              T5 = T5 + 1                                  : REM INCR COUNTER
6572  IF          T5 < T6                       THEN 6524      : REM CONT SEARCH
6576              T8$ = Q1$(M2)                                : REM STOR BIN PAG ADRS
6580  DATA LOAD   BA T #3, (D1,D0) Q1$()                       : REM RESET DISK INDX
6584  RETURN                                                   : REM EXIT
```

```
6588 REM **
6592 REM **
6596 REM **  Subroutine  ********************************************** 2 1 5
6600 REM **  Abstract >
6604 REM **    To assign a binary page address to return value.
6608 REM **    Assignment is according to function key index.
6612 REM **    Function index      1 - begining of index on first fiche
6616 REM **                        2 - begining of suggestions for
6620 REM **                            refresher training
6624 REM **                        3 - begining of practice excersices
6628 REM **                        4 - end of criterion tests
6632 REM **                        5 - diagram keyword (null function)
6636 REM **                        6 - extra table slot (null function)
6640 REM **                        7 - extra table slot (null function)
6644 REM **                        8 - extra table slot (null function)
6648 REM **  Variables >
6652 REM **    M1 > function key indx    ! T8$ > bin pag adrs (rturn value)
6656 REM **    Q4$() > adrs memory bufr !
6660 REM **
6664 REM ************************************************
6668  DEFFN ' 215 (M1,T8$)                                  : REM ENTRY
6672 REM **
6676 REM **        !   index  !   close  ! practice !    end    !
6680 REM **        !----------!----------!----------!-----------!
6684  ON M1 GOTO     6700    ,   6704    ,   6708    ,   6712    ,
                     6720    ,   6720    ,   6720    ,   6720
6688 REM **        !----------!----------!----------!-----------!
6692 REM **        ! diagram  !  extra   !   extra  !  extra    !
6696 REM **
6700                T8$ = Q4$(47)          : GOTO 6720 : REM INDX ADRS
6704                T8$ = Q4$(46)          : GOTO 6720 : REM CLOSE ADRS
6708                T8$ = Q4$(44)          : GOTO 6720 : REM PRACTICE ADRS
6712                T8$ = Q4$(48)          : GOTO 6720 : REM CRITRN END ADRS
6716 REM **
6720  RETURN                                             : REM EXIT
```

```
6724 REM **
6728 REM **
6732 REM **  Subroutine  ********************************************** 2 1 6
6736 REM **  Abstract >
6740 REM **    To save index frame on disk.
6744 REM **  Variables >
6748 REM **    M9 > function key index        ! L5$ > current bin pag adrs
6752 REM **    D7$ > converted pag addrs      ! W1$ > incrmnted pag adrs
6756 REM **    D1 > index sector adrs (curnt)! D2 > index bufr indx (curnt)
6760 REM **    L1 > index sector adrs (last) ! L2 > index bufr indx (last)
6764 REM **    D0 > scratch addrs register   !
6768 REM **
6772 REM ***********************************************
6776  DEFFN ' 216 (M9)                                    : REM ENTRY
6780  GOSUB ' 118 (L5$,D7$)                               : REM CONV PAG ADRS
6784  GOSUB ' 100 (1,59,4,D7$,0)                          : REM INSRT CURNT PAG
6788  IF          M9 = 0                        THEN 6816 : REM IF LAST PAGE
6792  GOSUB ' 103 (L5$,W1,W2,W3,1,3)                      : REM INCR PAGE ADDRS
6796              L5$ = W1$                               : REM STOR NEW PAG ADRS
6800  GOSUB ' 118 (W1$,D7$)                               : REM CONV PAGE ADDRS
6804  GOSUB ' 100 (16,41,23,"To Continue Go to ",0)       : REM INSRT CONT TEXT
6808  GOSUB ' 100 (16,59,4,D7$,0)                         : REM INSRT NXT PAG ADRS
6812  INIT(FF)    T8$()                                   : REM INIT KEY BUFFR
6816  GOSUB ' 206                                         : REM SAVE PAGE ON DISK
6820  IF          M9 = 0                        THEN 6868 : REM IF LAST PAGE
6824  GOSUB ' 101 (3,1,14,0)                              : REM ERASE SCREEN
6828  $TRAN       (Q2$()<129,896>, R4$()<177,4>)00 R      : REM CLR BUFFER
6832              D2 = D2 + 2                             : REM INCR INDX BUFR IDX
6836              L2 = L2 + 2                             : REM INCR INDX BUFR IDX
6840  IF          L2 < 128                      THEN 6852 : REM BUFR NOT EMPTY
6844              L1 = L1 + 1                             : REM INCR SCTR INDX
6848              L2 = 1                                  : REM RESET BUFR INDX
6852  IF          D2 < 128                      THEN 6868 : REM BUFR NOT EMPTY
6856              D1 = D1 + 1                             : REM INCR SCTR INDX
6860              D2 = 1                                  : REM RESET BUFR INDX
6864  DATA LOAD   BA T #3, (D1,D0) Q1$()                  : REM LOAD NEW INDX SCTR
6868  RETURN                                              : REM EXIT
```

```
6872 REM **
6876 REM **
6880 REM **  Subroutine   ********************************************  2 1 7
6884 REM **  Abstract >
6888 REM **    To Read the Data Subprogram and load into
6892 REM **    Storage buffer.
6896 REM **  Variables >
6900 REM **    C0 > initial bufr indx   ! C1 > row number
6904 REM **    C2 > col number          ! C3 > length of text
6908 REM **    C4 > control index       ! S4$() > input bufr
6912 REM **
6916 REM ************************************************
6920  DEFFN ' 217 (C0)                                    : REM ENTRY
6924  IF          C0 <> 3                      THEN 6940 : REM LOAD BUFR
6928  READ        C1,C2,C3,C4,S4$(1)                     : REM READ DATA PROGRAM
6932  IF          C4 <> 9                      THEN 6928 : REM NOT EOF
6936  GOTO        6956                                   : REM BRACNH TO EOF
6940  READ        C1,C2,C3,C4,S4$(1)                     : REM READ DATA PROGRAM
6944 REM **                                              : REM BRANCH
6948 REM **       ! lower ! caps ! bold ! line ! under !
6952 REM **       !-------!------!------!------!-------!
6956  ON C4 GOTO    6980  , 6992 , 7004 , 7040 , 7052  ,
                    7080  , 7080 , 7080 , 7092 , 7080
6960 REM **       !-------!------!------!------!-------!
6964 REM **       ! null  ! null ! null ! eof  ! null  !
6968 REM **
6972  GOTO        7080                                   : REM INSERT TEXT
6976 REM **
6980  $TRAN       (S4$() <1,C3>, R4$() <65,64>) R        : REM CONV TO LOWER CASE
6984  GOTO        7080                                   : REM INSERT TEXT
6988 REM **
6992  $TRAN       (S4$() <1,C3>, R4$() <1,64>)  R        : REM CONV TO UPPER CASE
6996  GOTO        7080                                   : REM INSERT TEXT
7000 REM **
7004              C0 = LEN(S4$(1))                       : REM SPECIAL PROCESS
7008  MAT COPY    -S4$() <1,C0> TO -S4$()<1,64>          : REM
7012  MAT SEARCH  S4$() <1,64>, <> R5$(1) TO R2$()       : REM
7016              C5 = FNR(VAL(R2$(1)))                  : REM
7020  MAT COPY    R5$() <2,1> TO S4$() <1,1>             : REM
7024  MAT COPY    S4$() <C5,C0> TO S4$() <2,63>          : REM
7028              C3 = C3 + 1                            : REM
7032  GOTO        7080                                   : REM INSERT TEXT
7036 REM **
7040  INIT        (SF) S4$()                             : REM CREATE LINE
7044  GOTO        7080                                   : REM INSERT TEXT
7048 REM **
7052              C4 = (C1-1)*64 + C2                    : REM CMPUTE BUF POS
7056  MAT COPY    Q2$() <C4,C3> TO S4$() <1,C3>          : REM LOAD FRM BUFR
7060  AND         (S4$(1),7F)                            : REM MASK TEXT
7064  ADD         (S4$(1),80)                            : REM UNDERSCORE
7068  MAT COPY    S4$() <1,C3>  TO Q2$() <C4,C3>         : REM RELOAD BUFR
7072  GOTO        7080                                   : REM INSERT TEXT
7076 REM **
7080  GOSUB ' 100 (C1,C2,C3,S4$(1),Q0(12))               : REM INSERT TEXT
7084  GOTO        6940                                   : REM GET NXT LINE
7088 REM **
7092  GOSUB ' 101 (1,1,0,0)                              : REM HOME CUSOR
7096  RETURN                                             : REM EXIT
```

AUTHOR System: Phase 3 - ATH3IDX2 - # MAP

```
6524  - 6564 6572
6540  - 6532
6548  - 6524
6576  - 6536
6700  - 6684
6704  - 6684
6708  - 6684
6712  - 6684
6720  - 6684 6700 6704 6708 6712
6816  - 6788
6852  - 6840
6868  - 6820 6852
6928  - 6932
6940  - 6924 7084
6956  - 6936
6980  - 6956
6992  - 6956
7004  - 6956
7040  - 6956
7052  - 6956
7080  - 6956 6972 6984 6996 7032 7044 7072
7092  - 6956
```

AUTHOR System: Phase 3 - ATH3IDX2 - V MAP

```
C0     - 6920 6924 7004 7008 7024
C1     - 6928 6940 7052 7080
C2     - 6928 6940 7052 7080
C3     - 6928 6940 6980 6992 7028 7056 7068 7080
C4     - 6928 6932 6940 6956 7052 7056 7068
C5     - 7016 7024
D0     - 6540 6580 6864
D1     - 6512 6580 6856 6864
D2     - 6516 6832 6852 6860
D7$    - 6780 6784 6800 6808
L1     - 6844
L2     - 6836 6840 6848
L5$    - 6780 6792 6796
M1     - 6512 6528 6532 6540 6668 6684
M2     - 6516 6524 6544 6548 6552 6576
M9     - 6776 6788 6820
Q0(    - 7080
Q1$(   - 6540 6552 6576 6580 6864
Q2$(   - 6828 7056 7068
Q4$(   - 6700 6704 6708 6712
Q9$(   - 6556 6560
R0$(   - 6560 6564
R2$(   - 7012 7016
R3$    - 6556
R4$(   - 6828 6980 6992
R5$(   - 7012 7020
S4$(   - 6928 6940 6980 6992 7004 7008 7012 7020 7024 7040 7056 7060 7064 7068
         7080
T5     - 6508 6568 6572
T6     - 6520 6572
T7     - 6468 6524 6528 6532 6544 6548
T8     - 6468 6520
T8$    - 6468 6576 6668 6700 6704 6708 6712
T8(    - 6472 6476 6480 6484 6488 6492 6496 6500 6504 6524 6528 6532 6544 6548
T8$(   - 6520 6812
W1     - 6792
W1$    - 6796 6800
W2     - 6792
W3     - 6792
```

```
AUTHOR System: Phase 3 - ATH3IDX2 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 100
       - 6784 6804 6808 7080
???? DEFFN' 101
       - 6824 7092
???? DEFFN' 103
       - 6792
???? DEFFN' 106
       - 6552
???? DEFFN' 118
       - 6780 6800
???? DEFFN' 206
       - 6816
6468 DEFFN' 214
6668 DEFFN' 215
6776 DEFFN' 216
6920 DEFFN' 217
```

5.33 ATH3LIST

The purpose of the 'ATH3LIST' program is to clear from memory all program text which is not needed for listing of the learning module. The program will also load in a new COMMON table.

The remainder of this section presents the 'ATH3LIST' program listing.

AUTHOR System: Phase 3 - ATH3LIST - COMCLR

```
1500 %## Program > AUTHOR/3  (ATH3LIST)    Version > 2.0  2200 T/VP  780323
1510 REM **  Abstract  ************************************************************
1520 REM **                                                                        **
1530 REM **    To output the Phse III data base onto selected                      **
1540 REM **    Output device.                                                      **
1550 REM **                                                                        **
1560 REM **    The listing program is loaded in 3 steps >                          **
1570 REM **      - Clear program text  (1500,9999)                                 **
1580 REM **      - Define Variable Table (100,300)                                 **
1590 REM **      - Load list program text (1500,9999)                              **
1600 REM **                                                                        **
1610 REM ************************************************************************
1620 REM **
1630 REM **                                                  CLEAR PROGRAM TEXT
1640 REM **
1650  COM CLEAR   B0                                        : REM CLR VAR TABLE
1660  LOAD DC T   #1, "ATH3LSTV" 100,300
1670  END                                                   : REM END OF STEP ONE
```

5.34 ATH3LSTV

The purpose of the 'ATH3LSTV' program is to define a new COMMON table and to load the program text needed to list/edit the learning module.

The remainder of this section presents the 'ATH3LSTV' program listing.

AUTHOR System: Phase 3 - ATH3LSTV - COMTBL

```
100  REM ** Program > AUTHOR/3  (ATH3LSTV)  Version > 2.0  2200 T/VP   780323
105  REM **  Abstract  ************************************************************
110  REM **                                                                      **
115  REM **    Step two of the list loading will define program                 **
120  REM **    variable storage.                                                 **
125  REM **                                                                      **
130  REM **                                                                      **
135  REM ************************************************************************
140  REM **
145  REM **
150  COM       R4$(20)16,R5$(10)1,R6$(16)1,R7$(64)1     : REM GENERAL REGISTERS
155  COM       R8$(10)24,R9$(1)64                       : REM GENERAL REGISTERS
160  COM       B0                                       : REM COM CLEAR POINTER
165  COM       D0,D1,D2,D3,D4,D5,D6,D7,D8,D9            : REM OUTPUT PAG INDX
170  COM       D4$4,D7$4                                : REM OUTPUT PAG INDX
175  COM       K1,K2,K3,K4,K5$(1)64                     : REM CUSUR CONTROL
180  COM       L1$25,L2$8,L3$4                          : REM PROGRAM LOADING
185  COM       W1,W2,W3,W4,W5,W6,W1$2                   : REM FICHE PAG INDX
190  COM       W2$16,W3$16,W4$16,W5$16,W6$16            : REM OUTPUT FLAGS
195  COM       R1,R2,R4,R5,R6,R8,R9                     : REM GENERAL REGISTERS
200  COM       R0$2,R2$64,R3$2,R7$64,R8$2,E1$4          : REM GENERAL BUFFERS
205  COM       R0$(1)1,R2$(2)2                          : REM GENERAL BUFFERS
210  COM       S1,S2,S3,S5,S4$(1)64                     : REM BUFR STORAGE INDX
215  COM       Q0(8),Q1$(128)2,Q2$(32,64)1              : REM GENERAL BUFFERS
220  DIM       Q9$(64)4                                 : REM SCRATCH BUFFER
225  %##            END OF COMMON STORAGE AREA          ##
230  REM **
235  REM **                                             LOAD LISTING PROGRAM
240  REM **
245  LOAD DC  T  #1, "ATH3LSTP" 1500,9999
300  END                                                : REM END OF STEP TWO
```

5.35 ATH3LSTP

The purpose of the 'ATH3LSTP' program is to allow the learning module to be displayed on one of three output devices. The program also controls the loading of the editing program.

The 'ATH3LSTP' program is presented in five parts.

5.35.1 ATH3LST1

This part of the 'ATH3LSTP' program contains the main control logic for controlling the list/edit process. The 'ATH3LST1' program also contains subroutines 108 to 111.

The remainder of this section presents the 'ATH3LST1' program listing and reference maps.

AUTHOR System: Phase 3 - ATH3LST1 - CNTRL

```
1500 %## Program > AUTHOR/3  (ATH3LST1)   Version > 2.0  2200 T/VP  780323
1504 REM **  Abstract  ************************************************
1508 REM **                                                            **
1512 REM **    To output the Phase 3 data base on to the selected      **
1516 REM **    output device.                                          **
1520 REM **                                                            **
1524 REM **    Currently avaiable devices are >                        **
1528 REM **      - CRT    <1>     <005>                                **
1532 REM **      - TTY    <2>     <011>                                **
1536 REM **      - LPT    <3>     <015>                                **
1540 REM **                                                            **
1544 REM ********************************************************************
1548 REM **
1552 REM **
1556  GOSUB ' 201  (W6)                    : REM SEL OUTPUT DEV
1560  GOSUB ' 202                          : REM INITIALIZE
1564  GOSUB ' 203                          : REM LOAD FRST/NXT PAG
1568  GOSUB ' 204                          : REM SCAN FOR KEYWORD
1572  GOSUB ' 210  (W6)                    : REM PRINT PAGE
1576  GOTO         1564                    : REM CONTINUE OUTPUT
```

```
1580 REM **
1584 REM ************************************************************************
1588 REM **                        LABELED SUBROUTINES                           **
1592 REM ************************************************************************
1596 REM **
1600 REM **  Subroutine  ********************************************** 1 0 8
1604 REM **    Function   >
1608 REM **      To Output page buffer onto the selected
1612 REM **      output device.
1616 REM **      Output index code       1 - CRT
1620 REM **                              2 - TTY
1624 REM **                              3 - LPT
1628 REM **                              4 - END OF LPT FILE
1632 REM **    Variables  >
1636 REM **      W6 > output device index     ! RO$() > input register
1640 REM **      W7 > undrscr lokup indx      ! W2$   > return scan flag
1644 REM **      W8 > bufer out lin indx      ! W3$   > return scan flag
1648 REM **                                   ! W4$   > return scan flag
1652 REM **
1656 REM ************************************************
1660  DEFFN ' 108 (W6)                                    : REM ENTRY
1664  GOSUB ' 124                                         : REM SEL SPL CHAR SCAN
1668 REM **
1672 REM **        !   CRT   !   TTY   !   LPT   !   END   !
1676 REM **        !---------!---------!---------!---------!
1680  ON W6 GOTO     1700   ,  1744   ,  1820   ,  1860    : REM BRANCH
1684 REM **        !---------!---------!---------!---------!
1688 REM **
1692 REM **                                                 OUTPUT TO CRT DEVICE
1696 REM **
1700  SELECT #5   005                                     : REM SEL OUTPUT DEV
1704  GOSUB ' 109                                         : REM CREATE CRT OVRLAY
1708  GOSUB ' 110                                         : REM SEL T OR VP DISPLY
1712  IF          W2$ = "VP SYSTEM"          THEN 1720    : REM VP SELECTED
1716  GOSUB ' 111                                         : REM DISPLAY OVRLAY
1720  GOSUB ' 112                                         : REM DISPLAY BUFR
1724  KEYIN       RO$(1), 1728, 1728       : GOTO 1712    : REM INPUT WAIT
1728  GOTO        1876                                    : REM BRANCH TO EXIT
1732 REM **
1736 REM **                                                 OUTPUT TO TTY DEVICE
1740 REM **
1744  SELECT #5   011                                     : REM SEL OUTPUT DEV
1748  GOSUB ' 109                                         : REM CREATE TTY OVRLAY
1752  GOSUB ' 113                                         : REM DISPLY INSTRUCTNS
1756  GOSUB ' 114 (1)                                     : REM PRINT UPR HASH
1760              W7 = 1                                  : REM INIT TABLE INDX 0
1764              X3 = 1                                  : REM INIT TABLE INDX 1
1768              Y3 = 1                                  : REM INIT TABLE INDX 2
1772  FOR         W8 = 1 TO 32                            : REM BEGIN BUFR LOOP
1776  GOSUB ' 115                                         : REM SCAN UNDRSCR TBL
1780  IF          W3$ = "NULL SCAN"          THEN 1792    : REM NO UNDERSCORING
1784  GOSUB ' 116                                         : REM PRINT UNDERSCORING
1788  GOTO        1796                                    : REM BRNCH TO LIN INCR
1792  GOSUB ' 117                                         : REM PRINT TEXT LINE
1796  NEXT        W8                                      : REM INCR LINE INDX
1800  GOSUB ' 114 (2)                                     : REM PRINT LWR HASH
1804  GOTO        1876                                    : REM BRANCH TO EXIT
```

```
1808 REM **
1812 REM **                                                  OUTPUT TO LPT DEVICE
1816 REM **
1820  SELECT #5   015                                      : REM SEL OUTPUT DEV
1824  SELECT      TAPE 07B                                 : REM SEL TAPE DEV
1828  GOSUB ' 118                                          : REM SEL UNDRSCR MODE
1832  GOSUB ' 119 (1)                                      : REM PRINT UPR HASH
1836  FOR         W8 = 1 TO 32                             : REM BEGIN BUFR LOOP
1840  GOSUB ' 117                                          : REM PRINT TEXT LINE
1844  NEXT        W8                                       : REM INCR LINE INDX
1848  GOSUB ' 119 (2)                                      : REM PRINT LOWR HASH
1852  IF          W4$ = "NO UNDERSCORE"      THEN 1864     : REM NO UNDERSCORING
1856  IF          W4$ = "OVERLAY"            THEN 1864     : REM IF OVRLAY OUTPUT
1860  GOSUB ' 120                                          : REM PRINT OVERLAY
1864  GOTO        1876                                     : REM BRANCH TO EXIT
1868 REM **
1872 REM **
1876  RETURN                                               : REM EXIT
```

```
1880 REM **
1884 REM **
1888 REM **  Subroutine  ********************************************** 1 0 9
1892 REM **  Abstract >
1896 REM **    To scan text buffer and create lookup table and
1900 REM **    overlay buffer for selected output devices.
1904 REM **  Variables >
1908 REM **    Q2$() > output text buffer     !
1912 REM **    W3$() > underscore lookup tbl ! W4$() > conversion buffer
1916 REM **    X4 > end of loop indx (row)    ! X5 > end of loop indx (col)
1920 REM **    R4 > lookup tbl indx (row)     ! R5 > lookup tbl indx (col)
1924 REM **    R6 > linear buff position      ! R5$() > single char storage
1928 REM **    W6 > output device indx        ! W9 > buffer size
1932 REM **
1936 REM ***********************************************
1940  DEFFN ' 109                                          : REM ENTRY
1944  DIM          W3$(1,200)2,W9$(7)8                     : REM DEFINE WORK BUFRS
1948  DIM          X3$(1,075)2,Y3$(1,075)2                 : REM DEFINE WORK BUFRS
1952               W9$(1) = HEX(5FA0)                      : REM INIT CONV VALUES
1956               W9$(2) = HEX(20800900)                  : REM INIT CONV VALUES
1960               W9$(3) = HEX(20802D812F825F83)          : REM INIT CONV VALUES
1964               W9$(4) = HEX(2000200120022003)          : REM INIT CONV VALUES
1968               W9$(5) = HEX(2020)                      : REM INIT CONV VALUES
1972               W9 = 2048                               : REM SET SIZE TO MAX
1976               R2$(1) = "Z"                            : REM DEFINE SPL CHAR 1
1980               R2$(2) = "O"                            : REM DEFINE SPL CHAR 2
1984  $TRAN        (Q2$()<1,W9>, W9$()<1,8>) R             : REM MINIZE UNDRSCRS
1988  IF           W6 <> 1                       THEN 2000 : REM NOT CRT OUTPUT
1992  MAT COPY     Q2$()<1,1024> TO Q2$()<1025,1024>       : REM SAVE UPPR BUFR
1996               W9 = 1024                               : REM SET BUFR SIZE
2000  MAT SEARCH   Q2$()<1,W9>, > R5$(3) TO W3$()          : REM CREAT LOKUP TBL 1
2004  IF           W6 = 1                        THEN 2012 : REM NOT TTY OUTPUT
2008  $TRAN        (Q2$()<1,W9>, W9$()<1,8>) 7F R          : REM CLR HIGH BIT
2012  MAT SEARCH   Q2$()<1,W9>, = R2$(1) TO X3$()          : REM CREAT LOKUP TBL 2
2016  MAT SEARCH   Q2$()<1,W9>, = R2$(2) TO Y3$()          : REM CREAT LOKUP TBL 3
2020               W3$(1,200)    = HEX(0000)             : REM SET EOI BYTE
2024               X3$(1,075)    = HEX(0000)             : REM SET EOI BYTE
2028               Y3$(1,075)    = HEX(0000)             : REM SET EOI BYTE
2032  IF           W6$ <> "NULL"                 THEN 2036 : REM NULL CHAR SCAN
2033               X3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
2034               Y3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
2035  GOTO         2112                                    : REM BRNCH TBL SCAN -
2036  IF           W6$ =  "BOTH"                 THEN 2048 : REM BEG Z&O TBL SCAN
2037  IF           W6$ =  "ZEE"                  THEN 2040 : REM BEG Z TBL SCAN
2038               X3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
2039  GOTO         2080                                    : REM BRNCH TBL SCAN O
2040               Y3$(1,1) = HEX(FFFF)                    : REM CLR LOCATOR BUFR
2044  REM **
2048  FOR          R4 = 1 TO 1                             : REM BEGIN TBL SCAN ROW
2052  FOR          R5 = 1 TO 075                           : REM BEGIN TBL SCAN COL
2056  GOSUB ' 125 (R4,R5,1,075,X3$(R4,R5),HEX(81))         : REM SCAN LOKUP TABLE 1
2060  NEXT         R5                                      : REM CONT  TBL SCAN COL
2064  NEXT         R4                                      : REM CONT  TBL SCAN ROW
2068               X3$(X6,X7) = HEX(FFFF)                  : REM SET EOI BYTE
2072  IF           W6$ = "BOTH"                  THEN 2084 : REM EXECUTE SCAN
```

```
2076  REM **
2080  IF          W6$ <> "ZERO"                THEN 2112 : REM NULL '0' SCAN
2084  FOR         R4 = 1 TO 1                            : REM BEGIN TBL SCAN ROW
2088  FOR         R5 = 1 TO 075                          : REM BEGIN TBL SCAN COL
2092  GOSUB ' 125 (R4,R5,1,075,Y3$(R4,R5),HEX(82))       : REM SCAN LOKUP TABLE 2
2096  NEXT        R5                                     : REM CONT  TBL SCAN COL
2100  NEXT        R4                                     : REM CONT  TBL SCAN ROW
2104              Y3$(X6,X7) = HEX(FFFF)                 : REM SET EOI BYTE
2108  REM **
2112  FOR         R4 = 1 TO 1                            : REM BEGIN TBL SCAN ROW
2116  FOR         R5 = 1 TO 200                          : REM BEGIN TBL SCAN COL
2120  GOSUB ' 125 (R4,R5,1,200,W3$(R4,R5),HEX(83))       : REM SCAN LOKUP TABLE 0
2124  NEXT        R5                                     : REM CONT  TBL SCAN COL
2128  NEXT        R4                                     : REM CONT  TBL SCAN ROW
2132              W3$(X6,X7) = HEX(FFFF)                 : REM SET EOI BYTE
2136  REM **
2140  REM **      !   CRT   !   TTY   !   LPT   !
2144  REM **      !---------!---------!---------!
2148  ON W6 GOTO     2160   ,  2172   ,  2184              : REM BRANCH
2152  REM **      !---------!---------!---------!
2156  REM **
2160  $TRAN       (Q2$()<1,W9>, W9$()<9,8>) 80 R         : REM CONV OVRLAY BUFR
2164  GOTO        2196                                   : REM BRANCH TO EXIT
2168  REM **
2172  $TRAN       (Q2$()<1,W9>, W9$()<1,8>) 7F R         : REM CLR UNDRSCR FLGS
2176  GOTO        2196                                   : REM BRANCH TO EXIT
2180  REM **
2184  $TRAN       (Q2$()<1,W9>, W9$()<17,24>) 83 R       : REM CONV OVRLAY BUFR
2188  GOTO        2196                                   : REM BRANCH TO EXIT
2192  REM **
2196  RETURN                                             : REM EXIT
```

```
2200 REM **
2204 REM **
2208 REM **  Subroutine  ********************************************** 1 1 0
2212 REM **  Abstract >
2216 REM **    To select type of CRT display to be used.
2220 REM **  Variables >
2224 REM **    W2$ > return type flag      ! RO$() > input register
2228 REM **
2232 REM ********************************************
2236  DEFFN ' 110                                              : REM ENTRY
2240  IF          W2$ = "VP SYSTEM"          THEN 2296 : REM SKIP ROUTINE
2244  IF          W2$ = "T  SYSTEM"          THEN 2296 : REM SKIP ROUTINE
2248              W2$ = "T  SYSTEM"                    : REM SET FLAG
2252  PRINT       HEX(03)                              : REM CLR CRT SCREEN
2256  GOSUB ' 101 (5,1,0,0)                            : REM POSITION CUSOR
2260  SELECT       PRINT 005                           : REM SEL OUTPUT MODE
2264  PRINTUSING   2312, " "                           : REM PRINT MENU
2268  PRINTUSING   2316, " "                           : REM PRINT MENU
2272  SELECT       PRINT 405                           : REM SEL OUTPUT MODE
2276  GOSUB ' 101 (2,1,0,0)                            : REM POSITION CUSOR
2280  PRINTUSING  2320, " ";                           : REM PRINT PROMPT
2284  KEYIN       RO$(1), 2288, 2288       : GOTO 2284 : REM RESPONCE WAIT
2288  IF          RO$(1) = "2"               THEN 2296 : REM SKP TO EXIT
2292              W2$ = "VP SYSTEM"                    : REM SET RTRUN FLAG
2296  PRINT       HEX(03)                              : REM CLR CRT SCREEN
2300 REM **
2304 REM **                                              OUTPUT IMAGE FORMATS
2308 REM **
2312  % CRT DISPLAY MENU      1 -  WANG 2200 VP Display   #
2316  %                       2 -  WANG 2200 T  Display   #
2320  %  Enter Type of CRT Display to be Used > #
2324 REM **
2328 REM **
2332  RETURN                                             : REM EXIT
```

```
2336 REM **
2340 REM **
2344 REM **  Subroutine  ********************************************** 1 1 1
2348 REM **  Abstract >
2352 REM **    To Display CRT overlay buffer onto selected CRT
2356 REM **    display.
2360 REM **  Variables >
2364 REM **    Q2$() > output ovrlay bufr    ! W9 > buffer out line indx
2368 REM **    R8$ > field format definition!
2372 REM **
2376 REM ***********************************************
2380  DEFFN ' 111                                          : REM ENTRY
2384             R8$ = HEX(5102)                          : REM DEFINE PACK FMT
2388  $GIO HOME  #5 (4001         ,R8$(9))                : REM HOME CUSOR
2392  FOR        W9 = 1 TO 16                             : REM BEG BUFFR OUT
2396  $PACK      (F=R8$) STR(R8$(9),7,2) FROM W9          : REM STOR LIN COUNT
2400  $GIO PRINT #5 (R8$(6),R8$(9))Q2$()<(W9-1)*64+1,64> : REM PRINT LINE
2404  NEXT       W9                                       : REM CUNT BUFR OUT
2408  RETURN                                              : REM EXIT
```

```
2412 REM **
2416 REM **
2420 REM **  Subroutine  ********************************************** 1 1 2
2424 REM **  Abstract >
2428 REM **    To Display CRT text buffer on to selected CRT
2432 REM **    display.
2436 REM **  Variables >
2440 REM **    Q2$() > overlay buffer        ! W9 > buffer out line indx
2444 REM **    R8$ > field format definition! R9$() > scratch buffer
2448 REM **
2452 REM ***********************************************
2456  DEFFN ' 112                                          : REM ENTRY
2460                R8$ = HEX(5102)                        : REM DEFINE PACK FMT
2464                R9$(1) = "Touch RETURN"                : REM INIT MESSAGE
2468  $GIO HOME     #5 (4001          ,R8$(9))             : REM HOME CUSUR
2472  FOR           W9 = 1 TO 16                           : REM BEG BUFFR OUT
2476  $PACK         (F=R8$) STR(R8$(9),7,2) FROM W9        : REM STOR LIN COUNT
2480                W8 = W9 + 16                           : REM CMPUT SCALE FACTR
2484  $GIO PRINT    #5 (R8$(6),R8$(9))Q2$()<((W8-1)*64+1,64> : REM PRINT LINE
2488  NEXT          W9                                     : REM CONT BUFR OUT
2492  $GIO MESG     #5 (400D400EA000,R8$(9))R9$()<1,13>    : REM PRINT MESSAGE
2496  RETURN                                               : REM EXIT
```

```
2500 REM **
2504 REM **
2508 REM **  Subroutine  ********************************************** 1 1 3
2512 REM **  Abstract >
2516 REM **    To Display Instructions to set up TTY device.
2520 REM **  Variables >
2524 REM **    W5$ > return scan flag      ! W4$() > i/o buffer #1
2528 REM **    W9  > scratch loop indx     ! W5$() > i/o buffer #2
2532 REM **    R0$() > input register      !
2536 REM **
2540 REM ***********************************************
2544  DEFFN ' 113                                          : REM ENTRY
2548  DIM          X9$(10)45                               : REM DEFINE WORK BUFRS
2552  IF           W5$ = "SKIP DISPLAY"       THEN 2696    : REM SKIP DISPLAY
2556               W5$ = "SKIP DISPLAY"                    : REM SET RETURN FLAG
2560  PRINT        HEX(03)                                 : REM CLR CRT SCREEN
2564  GOSUB ' 101  (5,5,0,0)                               : REM POSITION CUSOR
2568  PRINTUSING   2712, " "                               : REM PRINT INSTRUTN 1
2572  GOSUB ' 101  (7,5,0,0)                               : REM POSITION CUSOR
2576  PRINTUSING   2716, " "                               : REM PRINT INSTRUTN 2
2580  GOSUB ' 101  (9,5,0,0)                               : REM POSITION CUSOR
2584  PRINTUSING   2720, " "                               : REM PRINT INSTRUTN 3
2588  GOSUB ' 101  (16,1,0,0)                              : REM POSITION CUSOR
2592  PRINTUSING   2732, " ";                              : REM PRINT CONT PROMPT
2596  KEYIN        R0$(1), 2604, 2604      : GOTO 2596     : REM RESPONCE WAIT
2600 REM **
2604  $GIO READY   #5 (4500         ,R8$(9))               : REM SEND CTL CHAR
2608  $IF  ON      #5, 2632                                : REM DEVICE READY
2612  GOSUB ' 101  (11,10,0,0)                             : REM POSITION CUSOR
2616  PRINTUSING   2724, " "                               : REM PRINT ERR MSG
2620  $GIO PAUSE   /005(R8$(4),R8$(9))                     : REM ERROR TONE
2624  GOSUB ' 101  (11,10,1,0)                             : REM ERASE ERR MSG
2628  GOTO         2604                                    : REM CONT READY TEST
2632  PRINT        HEX(030A0A0A0A0A)                       : REM CLR CONSOLE DEV
2636  PRINTUSING   2728, " "                               : REM PRINT INSTRUCTN
2640 REM **
2644  GOSUB ' 121                                          : REM CLR/SET TAB STOPS
2648  SELECT       PRINT 211(100)                          : REM SEL CONSOLE DEVICE
2652  PRINTUSING   2736, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2656  PRINTUSING   2740, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2660  PRINTUSING   2744, " "          : PRINT HEX(0A0A)    : REM PRINT INSTRUCTNS
2664  PRINTUSING   2748, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2668  PRINTUSING   2752, " "          : PRINT HEX(0A0A)    : REM PRINT INSTRUCTNS
2672  PRINTUSING   2756, " ";                              : REM PRINT INSTRUCTNS
2676  PRINTUSING   2760, " "          : PRINT HEX(0A)      : REM PRINT INSTRUCTNS
2680  INIT(0A)     X9$()                                   : REM INIT FOR LINE FEED
2684  $GIO LF      #5 (A000         ,R8$(9))X9$()<1,7>     : REM ADJUST PAPER
2688  SELECT       PRINT 405(80)                           : REM SEL CONSOLE DEVICE
2692 REM **
2696  GOSUB ' 122                                          : REM DEVICE RESET
```

```
2700 REM **
2704 REM **                                                  INSTRUCITON FORMATS
2708 REM **
2712  % * * * * *  INSERT PAPER INTO TYPEWRITER  * * * *           #
2716  % * * * * *  SET LEFT MARGIN TO LEFT EDGE OF PAPER  * * * * *  #
2720  % * * * * *  SET SPACING SELECTOR TO SINGLE SPACING  * * * * * #
2724  %          * * * *  TYPEWRITER IS NOT READY  * * * *          #
2728  % * * * * *  FOLLOW INSTRUCTIONS ON OUTPUT DEVICE  * * * * * #
2732  %    Touch 'RETURN' When Completed >                          #
2736  %     #     AFTER EACH PAGE IS PRINTED THE PROGRAM WILL STOP.
2740  %     #     CHECK THAT THE PAPER IS PROPERLY ALIGNED...
2744  %     #     MAKE ADJUSTMENTS WHEN NESSARY.
2748  %     #     TO CONTINUE ... Switch the  'MANL/AUTO'  SWITCH TO
2752  %     #       'MANL'  POSITION THEN SWITCH TO  'AUTO' POSITION.
2756  %     # -=!----- ALIGN TOP EDGE OF PAPER TO THIS POSITION ...
2760  %     # THEN CONTINUE ------------!=-
2764 REM **
2768 REM **
2772  RETURN                                                 : REM EXIT
```

```
2776 REM **
2780 REM **
2784 REM **  Subroutine  ********************************************** 1 1 4
2788 REM **  Abstrct >
2792 REM **    To print upper/lower hash lines on the TTY device.
2796 REM **  Variables >
2800 REM **    W7$() > hash contorl buffer  ! W9 > upper/lower index
2804 REM **
2808 REM ********************************************
2812  DEFFN ' 114 (W9)                                    : REM ENTRY
2816  DIM          W7$(19)4,W8$(1)64
2820       W7$(1)=HEX(09210921) : W7$(2)=HEX(08080808): REM INIT TABLE
2824       W7$(3)=HEX(5F5F5F5F) : W7$(4)=HEX(0A0A0A0A): REM INIT TABLE
2828       W7$(5)=HEX(0A0A0A0A) : W7$(6)=HEX(0A0A0A0A): REM INIT TABLE
2832       W7$(7)=HEX(0A0A0A0A) : W7$(8)=HEX(0A0A0A0A): REM INIT TABLE
2836       W8$(1)="AUTHOR: An Automatic Authoring System      Version:  2.0"
2840 REM **
2844 REM **        !   upper    !   lower    !
2848 REM **        !------------!------------!
2852  ON W9 GOTO       2864     ,   2908                  : REM BRANCH
2856 REM **        !------------!------------!
2860 REM **
2864  $GIO TAB     #5 (40094020A000,R8$(9))W7$()<33,4> : REM HORZ TAB
2868  $GIO HEAD    #5 (A000        ,R8$(9))W8$()<1,56> : REM PRINT HEADING
2872  $GIO TOP     #5 (400D400A    ,R8$(9))            : REM VERTICAL TAB
2876  $GIO LINE 1 #5 (A0004000    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2880  $GIO LINE 2 #5 (A0004000    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2884  $GIO LINE 3 #5 (A000        ,R8$(9))W7$()<3,10> : REM PRINT HASH
2888  $GIO LINE 3 #5 (A000        ,R8$(9))W7$()<1,2>  : REM PRINT HASH
2892  $GIO LINE 3 #5 (A0004000    ,R8$(9))W7$()<8,5>  : REM PRINT HASH
2896  $GIO LF      #5 (A000        ,R8$(9))W7$()<13,3> : REM VERTICAL TAB
2900  GOTO         2940                                : REM BRANCH TO EXIT
2904 REM **
2908  $GIO LF      #5 (A000        ,R8$(9))W7$()<13,2> : REM VERTICAL TAB
2912  $GIO LINE 1 #5 (4009A000    ,R8$(9))W7$()<5,8>  : REM PRINT HASH
2916  $GIO LINE 1 #5 (4009A000400D,R8$(9))W7$()<9,4>  : REM PRINT HASH
2920  $GIO LINE 1 #5 (A0004000    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2924  $GIO LINE 2 #5 (A0004000    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2928  $GIO LINE 3 #5 (A0004000    ,R8$(9))W7$()<1,4>  : REM PRINT HASH
2932  $GIO LF      #5 (A000        ,R8$(9))W7$()<13,19>: REM VERTICAL TAB
2936 REM **
2940  RETURN                                           : REM EXIT
```

1564 - 1576
1700 - 1680
1712 - 1724
1720 - 1712
1728 - 1724
1744 - 1680
1792 - 1780
1796 - 1788
1820 - 1680
1860 - 1680
1864 - 1852 1856
1876 - 1728 1804 1864
2000 - 1988
2012 - 2004
2048 - 2036
2080 - 2040
2084 - 2072
2112 - 2032 2080
2160 - 2148
2172 - 2148
2184 - 2148
2196 - 2164 2176 2188
2284 - 2284
2288 - 2284
2296 - 2240 2244 2288
2312 - 2264
2316 - 2268
2320 - 2280
2596 - 2596
2604 - 2596 2628
2632 - 2608
2696 - 2552
2712 - 2568
2716 - 2576
2720 - 2584
2724 - 2616
2728 - 2636
2732 - 2592
2736 - 2652
2740 - 2656
2744 - 2660
2748 - 2664
2752 - 2668
2756 - 2672
2760 - 2676
2864 - 2852
2908 - 2852
2940 - 2900

```
Q2$(   - 1984 1992 2000 2008 2012 2016 2160 2172 2184 2400 2484
R0$(   - 1724 2284 2288 2596
R2$(   - 1976 1980 2012 2016
R4     - 2048 2056 2064 2084 2092 2100 2112 2120 2128
R5     - 2052 2056 2060 2088 2092 2096 2116 2120 2124
R5$(   - 2000
R8$    - 2384 2396 2460 2476
R8$(   - 2388 2396 2400 2468 2476 2484 2492 2604 2620 2684 2864 2868 2872 2876
         2880 2884 2888 2892 2896 2908 2912 2916 2920 2924 2928 2932
R9$(   - 2464 2492
W2$    - 1712 2240 2244 2248 2292
W3$    - 1780
W3$(   - 1944 2000 2020 2120 2132
W4$    - 1852 1856
W5$    - 2552 2556
W6     - 1556 1572 1660 1680 1988 2004 2148
W6$    - 2032 2036 2040 2072 2080
W7     - 1760
W7$(   - 2816 2820 2824 2828 2832 2864 2876 2880 2884 2888 2892 2896 2908 2912
         2916 2920 2924 2928 2932
W8     - 1772 1796 1836 1844 2480 2484
W8$(   - 2816 2836 2868
W9     - 1972 1984 1996 2000 2008 2012 2016 2160 2172 2184 2392 2396 2400 2404
         2472 2476 2480 2488 2812 2852
W9$(   - 1944 1952 1956 1960 1964 1968 1984 2008 2160 2172 2184
X3     - 1764
X3$(   - 1948 2012 2024 2056 2068
X6     - 2068 2104 2132
X7     - 2068 2104 2132
X9$(   - 2548 2680 2684
Y3     - 1768
Y3$(   - 1948 2016 2028 2092 2104
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 101
       - 2256 2276 2564 2572 2580 2588 2612 2624
1660 DEFFN' 108
1940 DEFFN' 109
       - 1704 1748
2236 DEFFN' 110
       - 1708
2380 DEFFN' 111
       - 1716
2456 DEFFN' 112
       - 1720
2544 DEFFN' 113
       - 1752
2812 DEFFN' 114
       - 1756 1800
???? DEFFN' 115
       - 1776
???? DEFFN' 116
       - 1784
???? DEFFN' 117
       - 1792 1840
???? DEFFN' 118
       - 1828
???? DEFFN' 119
       - 1832 1848
???? DEFFN' 120
       - 1860
???? DEFFN' 121
       - 2644
???? DEFFN' 122
       - 2696
???? DEFFN' 124
       - 1664
???? DEFFN' 125
       - 2056 2092 2120
???? DEFFN' 201
       - 1556
???? DEFFN' 202
       - 1560
???? DEFFN' 203
       - 1564
???? DEFFN' 204
       - 1568
???? DEFFN' 210
       - 1572
```

5.35.2 ATH3LST2

This part of the 'ATH3LSTP' program contains subroutines 115 to 123.

The remainder of the section presents the 'ATH3LST2' program listing and reference maps.

```
3000 %## Program > AUTHOR/3    (ATH3LST2)  Version > 2.0  2200 T/VP  780323
3004 REM **  Subroutine  ********************************************** 1 1 5
3008 REM **  Abstract >
3012 REM **    To scan lookup table for next underscored character.
3016 REM **  Variables >
3020 REM **    W7 > lookup table indx     ! W3$ > return scan flag
3024 REM **    W8 > text buffer lin indx ! R2$() > search return buffer
3028 REM **    W1 > next char position   ! W3$() > lookup table
3032 REM **
3036 REM **********************************************
3040  DEFFN ' 115                                          : REM ENTRY
3044  MAT COPY    W3$()<W7,2> TO R2$()<1,2>                : REM STOR BIN POSITION
3048              W1 = FNR(VAL(R2$(1)))                    : REM CMPUT BIN POSITION
3052  MAT COPY    X3$()<X3,2> TO R2$()<1,2>                : REM STOR BIN POSITION
3056              X4 = FNR(VAL(R2$(1)))                    : REM CMPUT BIN POSITION
3060  MAT COPY    Y3$()<Y3,2> TO R2$()<1,2>                : REM STOR BIN POSITION
3064              Y4 = FNR(VAL(R2$(1)))                    : REM CMPUT BIN POSITION
3068  REM **
3072  IF          W1 <= W8*64              THEN 3100       : REM SET POSITV SCAN
3076  IF          X4 <= W8*64              THEN 3100       : REM SET POSITV SCAN
3080  IF          Y4 <= W8*64              THEN 3100       : REM SET POSITV SCAN
3084  REM **
3088              W3$ = "NULL SCAN"                        : REM SET RETURN FLAG
3092  GOTO        3104                                     : REM BRANCH TO EXIT
3096 REM **
3100              W3$ = "POSITIVE SCAN"                    : REM SET RETURN FLAG
3104  RETURN                                               : REM EXIT
```

```
3108 REM **
3112 REM **
3116 REM **  Subroutine  ********************************************** 1 1 6
3120 REM **  Abstract >
3124 REM **    To print line containing underscoring on TTY device.
3128 REM **  Variables >
3132 REM **    W9$() > micro command buffr  ! W9 > temp text bufr position
3136 REM **    Q2$() > text buffer          ! W7 > lookup table indx
3140 REM **    W3$() > lookup table         ! W2 > next character position
3144 REM **    R2$() > search return bufr   ! W1 > curnt char position
3148 REM **    R8$() > micro command buffer ! W8 > text buffr lin indx
3152 REM **  Function Definitions >
3156  DEFFN    T(T) = T - (W9+WO-1)         : REM compute string length
3160 REM **
3164 REM ********************************************
3168  DEFFN ' 116                                      : REM ENTRY
3172  DIM          W9$(7)8                             : REM DEFINE WORK BUFR
3176               W9$(1) = HEX(40094020202020)        : REM INIT MICRO CMND
3180               W9$(2) = HEX(A0004008405F2020)      : REM INIT MICRO CMND
3184               W9$(3) = HEX(A000400D20202020)      : REM INIT MICRO CMND
3188               W9$(4) = HEX(A0004008402D2020)      : REM INIT MICRO CMND
3192               W9$(5) = HEX(A0004008402F2020)      : REM INIT MICRO CMND
3196 REM **
3200               WO     = 0                          : REM SET TEMP LIN INDX
3204               W9     = (W8-1)*64 + 1              : REM CMPUT BEG OF LINE
3208  $GIO TAB     #5 (W9$(1),R8$(9))                  : REM HORZ TAB
3212  REM **
3216  IF           W1 > X4                  THEN 3276 : REM W1 NOT MIN VALUE
3220  IF           W1 > Y4                  THEN 3296 : REM W1 NOT MIN VALUE
3224  $GIO PRINT   #5 (W9$(2),R8$(9))Q2$()<W9+WO,FNT(W1)>: REM PRINT STRING
3228  IF           W1 <> X4                 THEN 3252 : REM ALSO SPCL CHAR 1
3232  $GIO PRINT   #5 (W9$(4),R8$(9))Q2$()<W9+WO,0>    : REM PRINT SPCL CHAR 1
3236  GOSUB 3468                                       : REM INCR TBL INDX X3
3240  GOSUB 3400                                       : REM INCR TBL INDX W3
3244  GOTO         3312                                : REM BRNCH TO LIMIT CHK
3248  REM **
3252  IF           W1 <> Y4                 THEN 3264 : REM ALSO SPCL CHAR 2
3256  $GIO PRINT   #5 (W9$(5),R8$(9))Q2$()<W9+WO,0>    : REM PRINT SPCL CHAR 2
3260  GOSUB 3536                                       :  REM INCR TBL INDX Y3
3264  GOSUB 3400                                       :  REM INCR TBL INDX W3
3268  GOTO         3312                                : REM BRNCH TO LIMIT CHK
3272  REM **
3276  IF           X4 > Y4                  THEN 3296 : REM X4 NOT MIN VALUE
3280  $GIO PRINT   #5 (W9$(4),R8$(9))Q2$()<W9+WO,FNT(X4)>: REM PRINT STRING
3284  GOSUB 3468                                       : REM INCR TBL INDX X3
3288  GOTO  3312                                       : REM BRNCH TO LIMIT CHK
3292  REM **
3296  $GIO PRINT   #5 (W9$(5),R8$(9))Q2$()<W9+WO,FNT(Y4)>: REM PRINT STRING
3300  GOSUB 3536                                       : REM INCR TBL INDX Y3
3304  GOTO  3312                                       : REM BRNCH TO LIMIT CHK
3308  REM **
3312  IF           W1 <= W8*64              THEN 3216 : REM CONT THIS LINE
3316  IF           X4 <= W8*64              THEN 3216 : REM CONT THIS LINE
3320  IF           Y4 <= W8*64              THEN 3216 : REM CONT THIS LINE
```

```
3324  REM **
3328  MAT COPY     -Q2$()<W9,64> TO R9$()<1,64>          : REM REVERSE LIN COPY
3332  MAT SEARCH   R9$(), <> R5$(1) TO R2$()             : REM SRCH FOR NON BLNK
3336               W2 = FNR(VAL(R2$(1)))                 : REM CMPUT TRL SPCS
3340               W2 = (64-(W2-1)) - WO                 : REM CMPUT TRL SPACES
3344  $GIO END     #5 (W9$(3),R8$(9))Q2$()<W9+WO,W2>     : REM PRINT END OF LINE
3348  RETURN                                             : REM EXIT
```

```
3352 REM **
3356 REM **
3360 REM **  Subroutine  ********************************************** 1 1 6.1
3364 REM **  Abstract >
3368 REM **    This routine will increment and compute the index
3372 REM **    into the underscore lookup table.
3376 REM **  Variables >
3380 REM **    W7 > lookup table index  ! W3$() > lookup table
3384 REM **    W1 > compute position    ! W0 > temp bufr lin indx
3388 REM **    W9 > curnt begin of line ! R2$() > function input bufr
3392 REM **
3396 REM ***********************************************
3400              W7 = W7 + 2                          : REM INCR LOKUP TBL IDX
3404  MAT COPY    W3$()<W7,2> TO R2$()<1,2>            : REM FTCH NXT CHAR POS
3408              W0 = (W1+1) - W9                     : REM INCR TMP LIN INDX
3412              W1 = FNR(VAL(R2$(1)))                : REM CMPUT DEC CHAR POS
3416  RETURN                                           : REM EXIT
```

```
3420 REM **
3424 REM **
3428 REM **  Subroutine  ********************************************** 1 1 6.2
3432 REM **  Abstract >
3436 REM **    This routine will increment and compute the index
3440 REM **    into the 'Z' scan lookup table.
3444 REM **  Variables >
3448 REM **    X3 > lookup table index   ! X3$() > lookup table
3452 REM **    X4 > compute position     ! WO > temp bufr lin indx
3456 REM **    W9 > curnt begin of line ! R2$() > function input bufr
3460 REM **
3464 REM ***********************************************
3468                X3 = X3 + 2                                  : REM INCR LOKUP TBL IDX
3472  MAT COPY      X3$()<X3,2> TO R2$()<1,2>                    : REM FTCH NXT CHAR POS
3476                WO = (X4+1) - W9                             : REM INCR TMP LIN INDX
3480                X4 = FNR(VAL(R2$(1)))                        : REM CMPUT DEC CHAR POS
3484  RETURN                                                     : REM EXIT
```

```
3488 REM **
3492 REM **
3496 REM **  Subroutine  ********************************************** 1 1 6.3
3500 REM **  Abstract >
3504 REM **    This routine will increment and compute the index
3508 REM **    into the '0' Scan lookup table.
3512 REM **  Variables >
3516 REM **    Y3 > lookup table index  ! Y3$() > lookup table
3520 REM **    Y4 > compute position    ! WO > temp bufr lin indx
3524 REM **    W9 > curnt begin of line ! R2$() > function input bufr
3528 REM **
3532 REM ***********************************************
3536              Y3 = Y3 + 2                              : REM INCR LOKUP TBL IDX
3540  MAT COPY    Y3$()<Y3,2> TO R2$()<1,2>                : REM FTCH NXT CHAR POS
3544              WO = (Y4+1) - W9                         : REM INCR TMP LIN INDX
3548              Y4 = FNR(VAL(R2$(1)))                    : REM CMPUT DEC CHAR POS
3552  RETURN                                               : REM EXIT
```

```
3556 REM **
3560 REM **
3564 REM **  Subroutine  ********************************************** 1 1 7
3568 REM **  Abstract >
3572 REM **    To print a line without trailing blanks on selected
3576 REM **    output device.
3580 REM **  Variables >
3584 REM **    W9 > temp text bufr indx  ! Q2$() > text buffer
3588 REM **    R9$() > temp line bufr    ! R5$() > single char storage
3592 REM **    R2$() > serach retrn bufr ! W6 > output device indx
3596 REM **    WO > temp length of line  ! W7 > null line flag
3600 REM **
3604 REM ***********************************************
3608  DEFFN ' 117                                          : REM ENTRY
3612  DIM          W9$(7)8                                 : REM DEFINE WORK BUFR
3616               W9$(1) = HEX(5FA020202020)              : REM DEFINE CONVERSN
3620               W9 = (W8-1)*64 + 1                      : REM CMPUT BEG OF LIN
3624  $TRAN        (Q2$()<W9,64>, W9$()<1,8>) R            : REM MINIZE UNDERSCOR
3628  MAT COPY     -Q2$()<W9,64> TO R9$()<1,64>            : REM REVERSE COPY
3632  MAT SEARCH   R9$(), <> R5$(1) TO R2$()               : REM SCAN FOR NONBLANKS
3636  IF           R2$(1) <> HEX(0000)       THEN 3648     : REM TRAILING BLANKS
3640               BIN(STR(R2$(1),1,1)) = 0                : REM SET TO NULL LINE
3644               BIN(STR(R2$(1),2,1)) = 65               : REM SET TO NULL LINE
3648               WO = 64 - (FNR(VAL(R2$(1)))-1)          : REM CMPUT LEN OF STR
3652               BIN(STR(R8$(9),6,1)) = WO               : REM STOR LEN OF STR
3656               BIN(STR(R8$(9),7,1)) = 00               : REM SET LIMIT
3660  $GIO SETUP   #5 (R8$(8),R8$(9))                      : REM SETUP CR OR LF
3664 REM **
3668 REM **        !   CRT   !   TTY   !   LPT   !
3672 REM **        !---------!---------!---------!
3676  ON W6 GOTO     3724   ,  3688   ,  3704               : REM BRANCH
3680 REM **        !---------!---------!---------!
3684 REM **
3688  $GIO TAB     #5 (1467160840094020,R8$(9))            : REM HORZ TAB
3692  $GIO TEXT    #5 (A0004250     ,R8$(9))Q2$()<W9,WO>:  REM PRINT LINE
3696  GOTO         3724                                    : REM BRANCH TO EXIT
3700 REM **
3704  INIT(20)     R9$()                                   : REM CLR TAB BUFFER
3708  $GIO TAB     #5 (14671608A000,R8$(9))R9$()<1,4>      : REM HORZ TAB
3712  $GIO TEXT    #5 (A0004250     ,R8$(9))Q2$()<W9,WO>:  REM PRINT LINE
3716  GOTO         3724                                    : REM BRANCH TO EXIT
3720 REM **
3724  RETURN                                               : REM EXIT
```

```
3728 REM **
3732 REM **
3736 REM **  Subroutine  ********************************************** 1 1 8
3740 REM **  Abstract >
3744 REM **    To Display Instructions to set up LPT device.
3748 REM **  Variables >
3752 REM **    W4$ > return scan flag      ! W4$() > i/o buffer #1
3756 REM **    W9  > scratch loop indx     ! W5$() > i/o buffer #2
3760 REM **    RO$() > input register      !
3764 REM **
3768 REM ***********************************************
3772  DEFFN ' 118                                          : REM ENTRY
3776  DIM          X9$(10)45                               : REM DEFINE WORK BUFRS
3780  IF           W4$ = "NO UNDERSCORE"      THEN 3908    : REM SKIP ROUTINE
3784  IF           W4$ = "UNDERSCORE"         THEN 3908    : REM SKIP ROUTINE
3788  IF           W4$ = "OVERLAY"            THEN 3908    : REM SKIP ROUTINE
3792  IF           W4$ = "TAPE"               THEN 3908    : REM SKIP ROUTINE
3796  IF           W4$ = "SET PRINTER"        THEN 3856    : REM INIT PRINTER ONLY
3800               W4$ = "NO UNDERSCORE"                   : REM SET RETRN SCAN FLG
3804  PRINT        HEX(03)                                 : REM CLR CRT SCREEN
3808  GOSUB ' 101 (5,1,0,0)                                : REM POSITION CUSOR
3812  SELECT       PRINT 005                               : REM SEL OUTPUT MODE
3816  PRINTUSING   3888, " "                               : REM PRINT INSTRUTN 1
3820  PRINTUSING   3892, " "                               : REM PRINT INSTRUTN 2
3824  PRINTUSING   3896, " "                               : REM PRINT INSTRUTN 3
3828  SELECT       PRINT 405                               : REM SEL OUTPUT MODE
3832  GOSUB ' 101 (2,1,0,0)                                : REM POSITION CUSOR
3836  PRINTUSING   3900, " ";                              : REM PRINT PROMPT
3840  KEYIN        RO$(1), 3844, 3844       : GOTO 3840   : REM RESPONCE WAIT
3844  IF           RO$(1) <> "1"              THEN 3852   : REM SCAN FLAG SET
3848               W4$ = "UNDERSCORE"                      : REM SET SCAN FLAG
3852  IF           RO$(1) <> "2"              THEN 3868   : REM SCAN FLAG SET
3856               W4$ = "TAPE"                            : REM SET SCAN FLAG
3860  REM **
3864  GOSUB ' 127                                          : REM TEST TAPE READY
3868  GOSUB ' 128                                          : REM TEST PRINTER READY
3872  GOSUB ' 129                                          : REM PRINT INSTRUCTIONS
3876 REM **
3880 REM **                                                  INSTRUCITON FORMATS
3884 REM **
3888  % LPT UNDERSCORE MENU #  1 -  SINGLE PAGE UNDERSCORING
3892  %                        #  2 -  MULTI PAGE UNDERSCORING (TAPE)
3896  %                        #  3 -  DO NOT SCAN FOR UNDERSCORING
3900  % # Enter Code from LPT Underscoring Menu >
3904  REM **
3908  RETURN                                               : REM EXIT
```

```
3912 REM **
3916 REM **
3920 REM **  Subroutine  ********************************************** 1 1 9
3924 REM **  Abstrct >
3928 REM **    To print upper/lower hash lines on the LPT device.
3932 REM **  Variables >
3936 REM **    W7$() > hash contorl buffer  ! W9 > upper/lower index
3940 REM **
3944 REM ***********************************************
3948  DEFFN ' 119 (W9)                                          : REM ENTRY
3952  DIM          W7$(19)4,W8$(1)64
3956  INIT(20)     W7$()                                        : REM CLR TABLE
3960       W7$(1)=HEX(2D2D2D2D) : W7$(2)=HEX(0A0A0A0A): REM INIT TABLE
3964       W8$(1)="AUTHOR: An Automatic Authoring System      Version:  2.0"
3968 REM **
3972 REM **       !   upper   !   lower   !
3976 REM **       !-----------!-----------!
3980  ON W9 GOTO      3992    ,   4052                         : REM BRANCH
3984 REM **       !-----------!-----------!
3988 REM **
3992  $GIO TOP    #5 (400CA000     ,R8$(9))W7$()<5,3>  : REM TOP OF FORM
3996  $GIO TAB    #5 (A000         ,R8$(9))W7$()<9,8>  : REM HORZ TAB
4000  $GIO HEAD   #5 (A000400D     ,R8$(9))W8$()<1,64> : REM PRINT HEADING
4004  $GIO LF     #5 (A000         ,R8$(9))W7$()<5,3>  : REM VERTICAL TAB
4008  FOR         W9 = 1 TO 3                          : REM PRINT 3 LINES
4012  $GIO TAB    #5 (A000407C     ,R8$(9))W7$()<9,3>  : REM HORZ TAB/HASH
4016  $GIO TAB    #5 (A000407C400D,R8$(9))W7$()<9,64> : REM HORZ TAB/HASH
4020  NEXT        W9                                   : REM PRINT NXT LINE
4024  $GIO TAB    #5 (A000         ,R8$(9))W7$()<9,0>  : REM HORZ TAB
4028  $GIO HASH   #5 (A000         ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4032  $GIO TAB    #5 (A000         ,R8$(9))W7$()<9,64> : REM HORZ TAB
4036  $GIO HASH   #5 (A000400D     ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4040  $GIO LF     #5 (A000         ,R8$(9))W7$()<5,3>  : REM VERTICAL TAB
4044  GOTO        4092                                 : REM BRANCH TO EXIT
4048 REM **
4052  $GIO LF     #5 (A000         ,R8$(9))W7$()<5,3>  : REM VERTICAL TAB
4056  $GIO TAB    #5 (A000         ,R8$(9))W7$()<9,0>  : REM HORZ TAB
4060  $GIO HASH   #5 (A000         ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4064  $GIO TAB    #5 (A000         ,R8$(9))W7$()<9,64> : REM HORZ TAB
4068  $GIO HASH   #5 (A000400D     ,R8$(9))W7$()<1,4>  : REM PRINT HASH
4072  FOR         W9 = 1 TO 3                          : REM PRINT 3 LINES
4076  $GIO TAB    #5 (A000407C     ,R8$(9))W7$()<9,3>  : REM HORZ TAB/HASH
4080  $GIO TAB    #5 (A000407C400D,R8$(9))W7$()<9,64> : REM HORZ TAB/HASH
4084  NEXT        W9                                   : REM PRINT NXT LINE
4088 REM **
4092  RETURN                                           : REM EXIT
```

```
4096 REM **
4100 REM **
4104 REM **  Subroutine  ********************************************** 1 2 0
4108 REM **  Abstract >
4112 REM **    To print the underscore overlay buffer on to LPT device.
4116 REM **  Variables >
4120 REM **    W6 > output device indx    ! W4$ > return scan flag
4124 REM **    Z3 > end of file flag      !
4128 REM **
4132 REM ********************************************
4136  DEFFN ' 120                                          : REM ENTRY
4140  IF          W6 = 4                    THEN 4192    : REM END OF PRNT FILE
4144  GOSUB ' 109                                        : REM CREATE OVERLAY
4148  IF          W4$ = "TAPE"              THEN 4180    : REM WRITE ON TAPE
4152 REM **
4156  GOSUB ' 122                                        : REM DEVICE WAIT
4160              W4$ = "OVERLAY"                        : REM SET SCAN FLAG
4164  GOSUB ' 108 (W6)                                   : REM PRINT BUFFR
4168              W4$ = "UNDERSCORE"                     : REM RESET SCAN FLAG
4172  GOTO        4240                                   : REM BRANCH TO EXIT
4176 REM **
4180  GOSUB ' 190 (12,1,Z3,2048,1,6)                     : REM WRITE BUFFER
4184  GOTO        4240                                   : REM BRANCH TO EXIT
4188 REM **
4192  GOSUB ' 190 (10,3,Z3,0,1,6)                        : REM WRITE EOF
4196  DEFFN ' 016                                        : REM RECOVERY ENTRY
4200  GOSUB ' 190 (9,3,Z3,0,1,6)                         : REM REWIND TAPE
4204              W6 = 3                                 : REM RESET DEVICE INDX
4208  GOSUB ' 122                                        : REM DEVICE WAIT
4212              W4$ = "OVERLAY"                        : REM SET SCAN FLAG
4216  GOSUB ' 190 (7,1,Z3,2048,1,6)                      : REM READ RECORD
4220  IF          Z3 = 4                    THEN 4232    : REM END OF FILE
4224  GOSUB ' 108 (W6)                                   : REM PRINT BUFFER
4228  GOTO        4216                                   : REM READ NXT RECORD
4232  GOSUB ' 190 (9,2,Z3,0,1,6)                         : REM REWIND TAPE
4236 REM **
4240  RETURN                                             : REM EXIT
```

```
4244 REM **
4248 REM **
4252 REM **  Subroutine  ********************************************** 1 2 1
4256 REM **  Abstract >
4260 REM **    To Clear all tab stops and set the needed tab stops.
4264 REM **  Variables >
4268 REM **    W9 > scratch loop indx     ! R9$() > temp tab buffer
4272 REM **    W8 > scratch loop indx     !
4276 REM **
4280 REM ********************************************
4284  DEFFN ' 121                                          : REM ENTRY
4288 REM **
4292  FOR          W9 = 9 TO 1 STEP -4                     : REM DECRMNT LOOP
4296  $GIO CRLF    #5 (400D          ,R8$(9))              : REM CRLF
4300  FOR          W8 = 1 TO W9                            : REM TAB LOOP
4304  $GIO CLRTAB  #5 (40094019      ,R8$(9))              : REM TAB/CLEAR
4308  NEXT         W8                                      : REM NEXT TAB
4312  NEXT         W9                                      : REM DECRMNT LOOP
4316 REM **
4320  INIT(20)     R9$()                                   : REM CLR TAB BUFFR
4324  $GIO CRLF    #5 (400D          ,R8$(9))              : REM CRLF
4328  $GIO TABSET  #5 (A000401A      ,R8$(9))R9$()<1,18>   : REM TAB AND SET
4332  $GIO TABSET  #5 (4020A000401A,R8$(9))R9$()<1,64>     : REM TAB AND SET
4336  $GIO CRLF    #5 (400D          ,R8$(9))              : REM CRLF
4340 REM **
4344  RETURN                                               : REM EXIT
```

```
4348 REM **
4352 REM **
4356 REM **  Subroutine  ********************************************** 1 2 2
4360 REM **  Abstract >
4364 REM **    To allow the output device to be reset.  Program will
4368 REM **    pause until the device is disabled then will
4372 REM **    continue until a device ready is sensed.
4376 REM **  Variables >
4380 REM **    W9$() > micro command buffer  ! W6 > output device indx
4384 REM **
4388 REM ********************************************
4392  DEFFN ' 122                                          : REM ENTRY
4396  DIM          W9$(7)8                                 : REM DEFINE WORK BUFR
4400               W9$(1) = HEX(122141002020)              : REM INIT MICRO CMND
4404               W9$(2) = HEX(122145002020)              : REM INIT MICRO CMND
4408               W9$(3) = HEX(122141002020)              : REM INIT MICRO CMND
4412               W9$(4) = HEX(4465766963652020)          : REM INIT SCRN CHAR
4416               W9$(5) = HEX(2420202052656164)          : REM INIT SCRN CHAR
4420               W9$(6) = HEX(7920202020242020)          : REM INIT SCRN CHAR
4424               W9$(7) = HEX(5761697420202020)          : REM INIT SCRN CHAR
4428  $GIO TONE    /005 (R8$(4),R8$(9))                    : REM ATTENTION TONE
4432  GOSUB ' 123                                          : REM PRNT SCRN CHAR
4436  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
4440  $IF ON       #5,                              4428  : REM IF READY
4444  GOSUB ' 123                                          : REM PRNT SCRN CHAR
4448  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
4452  $IF ON       #5,                              4460  : REM IF READY
4456                                            GOTO 4444  : REM IF NOT READY
4460  $GIO TST ON #5    (W9$(W6),R8$(9))                   : REM SEND TST SIGNAL
4464  $IF ON       #5,                              4472  : REM IF READY
4468                                            GOTO 4444  : REM IF NOT READY
4472  RETURN                                               : REM EXIT
```

```
4476 REM **
4480 REM **
4484 REM **  Subroutine  ********************************************** 1 2 3
4488 REM **  Abstract >
4492 REM **    To Display a large cusor on console screen while
4496 REM **    waitting for device ready.
4500 REM **  Variables >
4504 REM **    W9$() > char storage bufr   ! W9 > temp bufr index (loop)
4508 REM **    R9$() > scratch cusor bufr  ! K1 > random row position
4512 REM **    K2 > random col position    !
4516 REM **
4520 REM ***********************************************
4524  DEFFN ' 123                                          : REM ENTRY
4528  INIT(08)     R9$()                                   : REM INIT CUSOR BUFR
4532               K1 = INT(RND(-1)*10)                    : REM GEN RND POS
4536  IF           K1 = 0                       THEN 4532  : REM INVALID POS
4540               K2 = INT(RND(-1)*100)                   : REM GEN RND POS
4544  IF           K2 > 55                      THEN 4540  : REM INVALID POS
4548  $GIO CLR     /005(4003          ,R8$(9))             : REM CLR CRT SCREEN
4552  GOSUB ' 101  (K1,K2,0,0)                             : REM POSITION CUSOR
4556  FOR          W9 = 25 TO 49 STEP 6                    : REM BEG CHAR PRNT
4560  $GIO PRINT   /005(4006A000     ,R8$(9))W9$()<W9,6>   : REM PRINT LINE
4564  $GIO BAKSPC  /005(4006400AA000,R8$(9))R9$()<1,6>     : REM POSITION FOR NXT
4568  NEXT         W9                                      : REM PRNT NXT LINE
4572  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH3LST2 - # MAP

```
3100  - 3072 3076 3080
3104  - 3092
3216  - 3312 3316 3320
3252  - 3228
3264  - 3252
3276  - 3216
3296  - 3220 3276
3312  - 3244 3268 3288 3304
3400  - 3240 3264
3468  - 3236 3284
3536  - 3260 3300
3648  - 3636
3688  - 3676
3704  - 3676
3724  - 3676 3696 3716
3840  - 3840
3844  - 3840
3852  - 3844
3856  - 3796
3868  - 3852
3888  - 3816
3892  - 3820
3896  - 3824
3900  - 3836
3908  - 3780 3784 3788 3792
3992  - 3980
4052  - 3980
4092  - 4044
4180  - 4148
4192  - 4140
4216  - 4228
4232  - 4220
4240  - 4172 4184
4428  - 4440
4444  - 4456 4468
4460  - 4452
4472  - 4464
4532  - 4536
4540  - 4544
```

```
K1      - 4532 4536 4552
K2      - 4540 4544 4552
Q2$(    - 3224 3232 3256 3280 3296 3328 3344 3624 3628 3692 3712
R0$(    - 3840 3844 3852
R2$(    - 3044 3048 3052 3056 3060 3064 3332 3336 3404 3412 3472 3480 3540 3548
          3632 3636 3640 3644 3648
R5$(    - 3332 3632
R8$(    - 3208 3224 3232 3256 3280 3296 3344 3652 3656 3660 3688 3692 3708 3712
          3992 3996 4000 4004 4012 4016 4024 4028 4032 4036 4040 4052 4056 4060
          4064 4068 4076 4080 4296 4304 4324 4328 4332 4336 4428 4436 4448 4460
          4548 4560 4564
R9$(    - 3328 3332 3628 3632 3704 3708 4320 4328 4332 4528 4564
T       - 3156
W0      - 3156 3200 3224 3232 3256 3280 3296 3340 3344 3408 3476 3544 3648 3652
          3692 3712
W1      - 3048 3072 3216 3220 3224 3228 3252 3312 3408 3412
W2      - 3336 3340 3344
W3$     - 3088 3100
W3$(    - 3044 3404
W4$     - 3780 3784 3788 3792 3796 3800 3848 3856 4148 4160 4168 4212
W6      - 3676 4140 4164 4204 4224 4436 4448 4460
W7      - 3044 3400 3404
W7$(    - 3952 3956 3960 3992 3996 4004 4012 4016 4024 4028 4032 4036 4040 4052
          4056 4060 4064 4068 4076 4080
W8      - 3072 3076 3080 3204 3312 3316 3320 3620 4300 4308
W8$(    - 3952 3964 4000
W9      - 3156 3204 3224 3232 3256 3280 3296 3328 3344 3408 3476 3544 3620 3624
          3628 3692 3712 3948 3980 4008 4020 4072 4084 4292 4300 4312 4556 4560
          4568
W9$(    - 3172 3176 3180 3184 3188 3192 3208 3224 3232 3256 3280 3296 3344 3612
          3616 3624 4396 4400 4404 4408 4412 4416 4420 4424 4436 4448 4460 4560
X3      - 3052 3468 3472
X3$(    - 3052 3472
X4      - 3056 3076 3216 3228 3276 3280 3316 3476 3480
X9$(    - 3776
Y3      - 3060 3536 3540
Y3$(    - 3060 3540
Y4      - 3064 3080 3220 3252 3276 3296 3320 3544 3548
Z3      - 4180 4192 4200 4216 4220 4232
```

```
AUTHOR System: Phase 3 - ATH3LST2 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
4196 DEFFN' 16
???? DEFFN' 101
       - 3808 3832 4552
???? DEFFN' 108
       - 4164 4224
???? DEFFN' 109
       - 4144
3040 DEFFN' 115
3168 DEFFN' 116
3608 DEFFN' 117
3772 DEFFN' 118
3948 DEFFN' 119
4136 DEFFN' 120
4284 DEFFN' 121
4392 DEFFN' 122
       - 4156 4208
4524 DEFFN' 123
       - 4432 4444
???? DEFFN' 127
       - 3864
???? DEFFN' 128
       - 3868
???? DEFFN' 129
       - 3872
???? DEFFN' 190
       - 4180 4192 4200 4216 4232
```

5.35.3 ATH3LST3

This part of the 'ATH3LSTP' program contains subroutines 124 to 129.

The remainder of this section presents the 'ATH3LST3' program listing.

```
4800 %## Program > AUTHOR/3  (ATH3LST3)  Version > 2.0  2200 T/VP  780323
4802 REM **
4804 REM **  Subroutine  ********************************************** 1 2 4
4806 REM **  Abstract >
4808 REM **    This routine will select the type of special
4810 REM **    character scanning to be executed.
4812 REM **  Variables >
4814 REM **    W6$ > scaning flag bufr ! RO$() > char input bufr
4816 REM **
4818 REM ************************************************
4820  DEFFN ' 124                                          : REM ENTRY
4822  IF          W6$ = "ZERO"              THEN 4906      : REM SCAN SELECTED
4824  IF          W6$ = "ZEE"               THEN 4906      : REM SCAN SELECTED
4826  IF          W6$ = "BOTH"              THEN 4906      : REM SCAN SELECTED
4828  IF          W6$ = "NULL"              THEN 4906      : REM SCAN SELECTED
4830  REM **
4832  PRINT       HEX(03)                                  : REM CLR CRT SCREEN
4834  GOSUB ' 101 (5,1,0,0)                                : REM POSITION CUSOR
4836  SELECT      PRINT 005                                : REM SEL DISPLY MODE
4838  PRINTUSING  4894, " "                                : REM PRINT MENU
4840  PRINTUSING  4896, " "
4842  PRINTUSING  4898, " "
4844  PRINTUSING  4900, " "
4846  SELECT      PRINT 405                                : REM SEL DISPLY MODE
4848  GOSUB ' 101 (2,1,0,0)                                : REM POSITION CUSOR
4850  PRINTUSING  4902, " "
4852  KEYIN       RO$(1), 4854, 4854       : GOTO 4852     : REM RESPONCE WAIT
4854  PRINT       HEX(03)                                  : REM CLR CRT SCREEN
4856              W6$ = "NULL"                             : REM SET DEFLT VALUE
4858  IF          RO$(1) <> "1"             THEN 4864      : REM SET '0' SCAN
4860              W6$ = "ZERO"
4862  REM **
4864  IF          RO$(1) <> "2"             THEN 4870      : REM SET 'Z' SCAN
4866              W6$ = "ZEE"
4868  REM **
4870  IF          RO$(1) <> "3"             THEN 4876      : REM SET '0-Z' SCAN
4872              W6$ = "BOTH"
4874  REM **
4876  IF          W6$ = "NULL"              THEN 4906      : REM BRNCH TO EXIT
4878  IF          W6 <> 3                   THEN 4906      : REM NOT LPT OUTPUT
4880              W4$ = "SET PRINTER"                      : REM SET SCAN FLAG
4882  SELECT #5   015                                      : REM SEL LINE PRINTER
4884  SELECT TAPE 07B                                      : REM SEL TAPE DRIVE
4886  GOSUB ' 118                                          : REM INIT PRINTER
4888  REM **
4890  REM **                                                 OUTPUT IMAGE FORMATS
4892  REM **
4894  % # Special Character Scanning >   1 - Scan for zero (0 <- /)
4896  % #                               2 - Scan for zee  (Z <- -)
4898  % #                               3 - Scan for both '0' & 'Z'
4900  % #                               4 - NO Special Scanning
4902  % # Enter Scanning Code from Below Table >
4904  REM **
4906  RETURN                                               : REM EXIT
```

```
4908 REM **
4910 REM **
4912 REM **  Subroutine  ********************************************** 1 2 5
4914 REM **  Abstract >
4916 REM **    This routine will scan search search return buffer
4918 REM **    and insert special character into text buffer in
4920 REM **    location specified by buffer value.
4922 REM **  Variables >
4924 REM **    R4 > lookup tbl row indx   ! R5 > lookup tbl col indx
4926 REM **    X6 > rtrn row indx on 0    ! X7 > rtrn col indx on 0
4928 REM **    X8 > max table row indx    ! X9 > max table col indx
4930 REM **    R2$() > search bufr value  ! R0$() > spcl insert char
4932 REM **    R6 > decimal bufr value    !
4934 REM **
4936 REM ************************************************
4938  DEFFN ' 125 (R4,R5,X8,X9,R2$(1),R0$(1))          : REM ENTRY
4940              R6 = FNR(VAL(R2$(1)))                : REM CMPUT DECMAL POS
4942  IF          R6 <> 0                  THEN 4952  : REM NOT END OF TABLE
4944              X6 = R4     :    X7 = R5             : REM SAVE END OF TABLE
4946              R4 = X8     :    R5 = X9             : REM SEND END OF LOOP
4948  GOTO 4956                                       : REM BRNCH TO EXIT
4950  REM **
4952  IF          W6 = 2                   THEN 4956  : REM TTY OUTPUT
4954  MAT COPY    R0$()<1,1> TO Q2$()<R6,1>           : REM INSRT SPCL CHAR
4956  RETURN                                          : REM EXIT
```

```
4958 REM **
4960 REM **
4962 REM **  Subroutine  ********************************************** 1 2 6
4964 REM **  Abstract >
4966 REM **    This routine will print the text string buffer
4968 REM **    X9$() as specified by input string parameters.
4970 REM **  Variables >
4972 REM **    X8 > beging string position ! X9 > length of string
4974 REM **
4976 REM ********************************************
4978  DEFFN ' 126 (X8,X9)                                : REM ENTRY
4980  $GIO PRINT  #5 (A000    ,R8$(9))X9$()<X8,X9>       : REM PRINT TEXT STRING
4982  RETURN                                             : REM EXIT
```

```
4984 REM **
4986 REM **
4988 REM **  Subroutine  ********************************************** 1 2 7
4990 REM **  Abstract >
4992 REM **    This routine will test the tape unit for ready status.
4994 REM **  Variables >
4996 REM **    W4$ > underscor status flag ! W6$ > special char scan flag
4998 REM **    W6 > output device flag     !
5000 REM **
5002 REM ************************************************
5004  DEFFN ' 127                                          : REM ENTRY
5006  $GIO TST ON (6CF5440086077105122140O0,R8$(9))       : REM SEND TEST SIGNAL
5008  $1F      ON /07B, 5042                               : REM TAPE READY
5010              W4$ = " "                                : REM CLR SCAN FLAG
5012              W6$ = " "                                : REM CLR SCAN FLAG
5014              W6  = 8                                  : REM SET FOR ERR RECVRY
5016  GOSUB ' 101 (11,10,0,0)                             : REM POSITION CUSOR
5018  PRINTUSING  5036, " "                               : REM PRINT ERR MSG
5020  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
5022  GOSUB ' 101 (13,1,0,0)                              : REM POSITION CUSOR
5024  PRINTUSING  5038, " "                               : REM PRINT ERR MSG
5026  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
5028  GOTO        5008                                    : REM CONT TEST
5030  REM **                                                OUTPUT IMAGE FORMATS
5032  REM **
5034  REM **
5036  % # * * * *   TAPE UNIT NOT READY   * * * *
5038  % # Touch 'RESET'... Then Type   RUN 1500   and Touch 'RETURN'
5040  REM **
5042  RETURN                                               : REM EXIT
```

```
5044 REM **
5046 REM **
5048 REM **  Subroutine  ********************************************** 1 2 8
5050 REM **  Abstract >
5052 REM **    This routine will test the printer for ready status.
5054 REM **  Variables >
5056 REM **    R8$() > i/o micro command buffer !
5058 REM **
5060 REM ***********************************************
5062  DEFFN ' 128                                          : REM ENTRY
5064  $CIO READY  #5 (4100         ,R8$(9))               : REM SEND CTL CHAR
5066  $1F  ON     #5, 5078                                : REM DEVICE READY
5068  GOSUB ' 101 (11,10,0,0)                             : REM POSITION CUSOR
5070  PRINTUSING  5088, " "                               : REM PRINT ERR MSG
5072  $CIO PAUSE  /005(R8$(4),R8$(9))                     : REM ERROR TONE
5074  GOSUB ' 101 (11,10,1,0)                             : REM ERASE ERR MSG
5076  GOTO        5064                                    : REM CONT READY TEST
5078  PRINT       HEX(030A0A0A0A0A0A)                     : REM CLR CONSOLE DEV
5080  PRINTUSING  5090, " "                               : REM PRINT INSTRUCTN
5082  REM **
5084  REM **                                                OUTPUT IMAGE FORMATS
5086  REM **
5088  %          * * * *  LINEPRINTER IS NOT READY  * * * *       #
5090  % * * * *  FOLLOW INSTRUCTIONS ON OUTPUT DEVICE  * * * *   #
5092  REM **
5094  RETURN                                               : REM EXIT
```

```
5096 REM **
5098 REM **
5100 REM **  Subroutine  ********************************************** 1 2 9
5102 REM **  Abstract >
5104 REM **    This routine will display the alignment instructions
5106 REM **    on the lineprinter.
5108 REM **  Variables >
5110 REM **    X7 > temp text bufr indx  ! X9$() > temp text bufr
5112 REM **    R0$() > char input bufr   !
5114 REM **
5116 REM ***********************************************
5118  DEFFN ' 129                                        : REM ENTRY
5120  GOSUB 5248                                         : REM INIT TEXT BUFFR 1
5122  GOSUB ' 126 ( 34,12)                               : REM VERT TAB
5124  GOSUB ' 126 ( 46,30)                               : REM LEFT ARROWS
5126  GOSUB ' 126 (361, 5)                               : REM HORZ TAB
5128  GOSUB ' 126 (136,30)                               : REM TEXT
5130  GOSUB ' 126 ( 91,30)                               : REM RIGHT ARROWS
5132  GOSUB ' 126 ( 91,35)                               : REM RIGHT ARROWS
5134  GOSUB ' 126 ( 44, 2)                               : REM CRLF
5136  REM **
5138  GOSUB ' 126 (181,30)                               : REM LEFT MARGIN
5140  GOSUB ' 126 (226,40)                               : REM SIZE
5142  GOSUB ' 126 (361,35)                               : REM HORZ TAB
5144  GOSUB ' 126 (316,25)                               : REM RIGHT MARGIN
5146  GOSUB ' 126 ( 44, 2)                               : REM CRLF
5148  REM **
5150  GOSUB ' 126 (361,0)                                : REM HORZ TAB
5152  GOSUB ' 126 (181,30)                               : REM LEFT MARGIN
5154  GOSUB ' 126 (271,20)                               : REM SIZE
5156  GOSUB ' 126 (316,25)                               : REM RIGHT MARGIN
5158  GOSUB ' 126 ( 44, 2)                               : REM CRLF
5160  REM **
5162  REM **
5164  GOSUB 5284                                         : REM INIT TEXT BUFFR 2
5166  GOSUB ' 126 (285,10)                               : REM HORZ TAB
5168  GOSUB ' 126 (406,40)                               : REM HORZ LINE
5170  GOSUB ' 126 (406,40)                               : REM HORZ LINE
5172  GOSUB ' 126 ( 44, 2)                               : REM CRLF
5174  REM **
5176  FOR         X7 = 2 TO 9                            : REM BEG LINE LOOP
5178  GOSUB ' 126 (285,30)                               : REM HORZ TAB
5180  GOSUB ' 126 (((X7-1)*45)+1,LEN(X9$(X7)))           : REM TEXT
5182  GOSUB ' 126 ( 44, 2)                               : REM CRLF
5184  NEXT        X7                                     : REM CONT LINE LOOP
5186  REM **
5188  GOSUB ' 126 (285,10)                               : REM HORZ TAB
5190  GOSUB ' 126 (406,40)                               : REM HORZ LINE
5192  GOSUB ' 126 (406,40)                               : REM HORZ LINE
5194  GOSUB ' 126 ( 44, 2)                               : REM CRLF
```

```
5196  REM **
5198  REM **
5200  GOSUB 5324                                          : REM INIT TEXT BUFFR 3
5202  FOR         X7 = 2 TO 8                             : REM BEGIN LINE LOOP
5204  GOSUB ' 126 (361,30)                                : REM HORZ TAB
5206  GOSUB ' 126 (((X7-1)*45)+1,LEN(X9$(X7)))            : REM TEXT
5208  GOSUB ' 126 ( 44, 2)                                : REM CRLF
5210  NEXT        X7                                      : REM CONT LINE LOOP
5212  REM **
5214  REM **
5216  $GIO TOP    #5 (400C     ,R8$(9))                   : REM ISSUE FORM FEED
5218  KEYIN       RO$(1), 5228, 5218                      : REM RESPONCE WAIT
5220  IF          RO$(1) = HEX(81)          THEN 5218     : REM NO TONE
5222  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM PAUSE TONE
5224  GOTO        5218                                    : REM CONTINUE WAIT
5226  REM **
5228  IF          RO$(1) = HEX(84)          THEN 5120     : REM REPRINT PAGE
5230  IF          RO$(1) = HEX(81)          THEN 5218     : REM SHUT OFF TONE
5232  RETURN                                              : REM EXIT
```

```
5234 REM **
5236 REM **
5238 REM **  Subroutine  ********************************************** 1 2 9.1
5240 REM **  Abstract >
5242 REM **    This routine will load text buffer with text group 1
5244 REM **
5246 REM ************************************************
5248  INIT (20)    X9$()                                  : REM LOAD TEXT BUFR 1
5250  INIT (0A)    X9$(01)
5252  INIT (0D)    STR(X9$(1),45,1)
5254  INIT ("<")   X9$(02)
5256  INIT (">")   X9$(03)
5258               X9$(04) = "Align to Sixth Paper Hole"
5260               X9$(05) = "<<<<<------- Left Margin"
5262               X9$(06) = "14 7/8  X  11"
5264               X9$(07) = "08 1/2  X  11"
5266               X9$(08) = "Right Margin ------->>>>>"
5268  RETURN                                              : REM EXIT
```

```
5270  REM **
5272  REM **
5274 REM **  Subroutine  ********************************************** 1 2 9.2
5276 REM **  Abstract >
5278 REM **    This routine will load text buffer with text group 2
5280 REM **
5282 REM ***********************************************
5284  INIT (20)   X9$()                                  : REM LOAD TEXT BUFR 2
5286  INIT (0A)   X9$(1)
5288  INIT (0D)   STR(X9$(1),45,1)
5290  INIT (2D)   X9$(10)
5292              X9$(2)  = "This page should now be aligned"
5294              X9$(3)  = "to the desired position..."
5296              X9$(4)  = "--  To Continue to print THIS PAGE"
5298              X9$(5)  = "    TOUCH the 'CONTINUE' Key."
5300              X9$(6)  = "--  To Begin Listing TOUCH the"
5302              X9$(7)  = "    'RUN' Key."
5304              X9$(8)  = "--  This page should appear on"
5306              X9$(9)  = "    an UPWARD Facing Page."
5308  RETURN                                           : REM EXIT
```

```
AUTHOR System: Phase 3 - ATH3LST3 -1 2 9 .3

5310  REM **
5312  REM **
5314 REM **  Subroutine  ********************************************** 1 2 9.3
5316 REM **  Abstract >
5318 REM **    This routine will load text buffer with text group 3
5320 REM **
5322 REM ***********************************************
5324  INIT (20)    X9$()                                  : REM LOAD TEXT BUFR 3
5326  INIT (0A)    X9$(1)
5328  INIT (0D)    STR(X9$(1),45,1)
5330               X9$(2)  = "Special Instructions for Underscoring:"
5332               X9$(3)  = "--  Single Page Overlaying:  At the"
5334               X9$(4)  = "    sound of the tone, position page"
5336               X9$(5)  = "    two inches above top of form."
5338               X9$(6)  = "--  Multi Page Overlaying:  At the"
5340               X9$(7)  = "    sound of the tone, position paper"
5342               X9$(8)  = "    to this page and position."
5344  RETURN                                              : REM EXIT
```

AUTHOR System: Phase 3 - ATH3LST3 - # MAP

```
4852  - 4852
4854  - 4852
4864  - 4858
4870  - 4864
4876  - 4870
4894  - 4838
4896  - 4840
4898  - 4842
4900  - 4844
4902  - 4850
4906  - 4822 4824 4826 4828 4876 4878
4952  - 4942
4956  - 4948 4952
5008  - 5028
5036  - 5018
5038  - 5024
5042  - 5008
5064  - 5076
5078  - 5066
5088  - 5070
5090  - 5080
5120  - 5228
5218  - 5218 5220 5224 5230
5228  - 5218
5248  - 5120
5284  - 5164
5324  - 5200
```

```
Q2$(   - 4954
R0$(   - 4852 4858 4864 4870 4938 4954 5218 5220 5228 5230
R2$(   - 4938 4940
R4     - 4938 4944 4946
R5     - 4938 4944 4946
R6     - 4940 4942 4954
R8$(   - 4980 5006 5020 5026 5064 5072 5216 5222
W4$    - 4880 5010
W6     - 4878 4952 5014
W6$    - 4822 4824 4826 4828 4856 4860 4866 4872 4876 5012
X6     - 4944
X7     - 4944 5176 5180 5184 5202 5206 5210
X8     - 4938 4946 4978 4980
X9     - 4938 4946 4978 4980
X9$(   - 4980 5180 5206 5248 5250 5252 5254 5256 5258 5260 5262 5264 5266 5284
         5286 5288 5290 5292 5294 5296 5298 5300 5302 5304 5306 5324 5326 5328
         5330 5332 5334 5336 5338 5340 5342
```

```
AUTHOR System: Phase 3 - ATH3LST3 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 101
       - 4834 4848 5016 5022 5068 5074
???? DEFFN' 118
       - 4886
4820 DEFFN' 124
4938 DEFFN' 125
4978 DEFFN' 126
       - 5122 5124 5126 5128 5130 5132 5134 5138 5140 5142 5144 5146 5150 5152
         5154 5156 5158 5166 5168 5170 5172 5178 5180 5182 5188 5190 5192 5194
         5204 5206 5208
5004 DEFFN' 127
5062 DEFFN' 128
5118 DEFFN' 129
```

5.35.4 ATH3LST4

This part of the 'ATH3LSTP' program contains subroutines 19Ø and 191.

The remainder of this section presents the 'ATH3LST4' program listing and reference maps.

```
5500 %## Program > AUTHOR/3  (ATH3LST4)  Version > 2.0 2200 T/VP  780323
5503 REM **
5506 REM **
5509 REM **  Subroutine  ********************************************** 1 9 0
5512 REM **  Abstract >
5515 REM **    This is a general nine track utility package. This
5518 REM **    Routine will perform the following functions >
5521 REM **      1 - <  BSF  > - Backspace one File;
5524 REM **      2 - <  BSR  > - Backspace one Record (read ready);
5527 REM **      3 - <  BSW  > - Backspace one Record (write ready);
5530 REM **      4 - < CLEAN> - Clean local area on tape;
5533 REM **      5 - <  FSF  > - Space Forward one File;
5536 REM **      6 - <  FSR  > - Space Forward one Record;
5539 REM **      7 - < READ  > - Read Record;
5542 REM **      8 - <REREAD> - Backspace and Read Record;
5545 REM **      9 - <REWIND> - Rewind tape to Load point;
5548 REM **     10 - < WEOF  > - Write End Of File mark;
5551 REM **     11 - < WGAP  > - Write Inter Record Gap;
5554 REM **     12 - < WRITE> - Write Record.
5557 REM **    A scan for I/O errors will be performed according to
5560 REM **    the following index >
5563 REM **      1 - Data Input/Output errors  (7,12,8)
5566 REM **      2 - Position errors           (1,2,5,6,9,4)
5569 REM **      3 - Control Output errors     (10,11,3)
5572 REM **      4 - No error processing
5575 REM **    A error index is returned in variable >  Z3
5578 REM **    Values of Z3 are >       0  -  Fatal Error
5581 REM **                             1  -  Tape at Load Point
5584 REM **                             2  -  Tape is Write Disabled
5587 REM **                             3  -  End of Tape Sensed
5590 REM **                             4  -  End of File Sensed
5593 REM **  Variables >
5596 REM **    Z1 > Function index         ! Z2 > error index
5599 REM **    Z3 > return error index     ! Z4 > transfer buffer size
5602 REM **    Z5 > suppress error mesg    ! Z6 > device table indx
5605 REM **    Z1$() > micro cmnd buffer   ! Z2$() > error mask buffer
5608 REM **    W2$() > i/o buffer          !
5611 REM **
5614 REM ************************************************
```

```
5617  DEFFN ' 190 (Z1,Z2,Z3,Z4,Z5,Z6)                          : REM ENTRY
5620  DIM         Z1$(13)10,Z2$(12)1,Z3$40                     : REM DEFINE WORK BUFRS
5623  DEFFN       Z(Z)=256*VAL(STR(Z1$(Z),9,1))+FNY(Z):        REM DEFINE FUNCTION
5626  DEFFN       Y(Y)=VAL(STR(Z1$(Y),10,1))                   : REM DEFINE FUNCTION
5629  SELECT      TAPE 07B                                     : REM SEL TAPE DEVICE
5632    Z1$( 1) = HEX(6CF34400B607)                            : REM DEFINE MICRO CMND
5635    Z1$( 2) = HEX(6CF14400B607)                            : REM DEFINE MICRO CMND
5638    Z1$( 3) = HEX(6CF04400B607)                            : REM DEFINE MICRO CMND
5641    Z1$( 4) = HEX(6CF74400B607)                            : REM DEFINE MICRO CMND
5644    Z1$( 5) = HEX(6CF44400B607)                            : REM DEFINE MICRO CMND
5647    Z1$( 6) = HEX(6CF24400B607)                            : REM DEFINE MICRO CMND
5650    Z1$( 7) = HEX(6CFB4400C2228607)                        : REM DEFINE MICRO CMND
5653    Z1$( 8) = HEX(6CF84400C2228607)                        : REM DEFINE MICRO CMND
5656    Z1$( 9) = HEX(6CF54400B607)                            : REM DEFINE MICRO CMND
5659    Z1$(10) = HEX(6CF94400B607)                            : REM DEFINE MICRO CMND
5662    Z1$(11) = HEX(6CF64400B607)                            : REM DEFINE MICRO CMND
5665    Z1$(12) = HEX(6CFA4400A2068607)                        : REM DEFINE MICRO CMND
5668    Z1$(13) = HEX(1122334455660000000)                     : REM DEFINE MICRO CMND
5671 REM **
5674    Z2$( 1) = HEX(80)    :    Z2$( 2) = HEX(40)            : REM DEFINE ERR MASKS
5677    Z2$( 3) = HEX(20)    :    Z2$( 4) = HEX(10)            : REM DEFINE ERR MASKS
5680    Z2$( 5) = HEX(08)    :    Z2$( 6) = HEX(04)            : REM DEFINE ERR MASKS
5683    Z2$( 7) = HEX(02)    :    Z2$( 8) = HEX(04)            : REM DEFINE ERR MASKS
5686    Z2$( 9) = HEX(02)    :    Z2$(10) = HEX(01)            : REM DEFINE ERR MASKS
5689 REM **
5692  $GIO TAPE   (Z1$(Z1),Z1$(13))   Q2$()<1,Z4>               : REM EXECUTE FUNCTION
5695 REM **
5698              Z3 = 0                                       : REM CLR RETURN FLAG
5701 REM **
5704 REM **       !   I/O  !   POS  !   CTL  !  NULL  !
5707 REM **       !--------!--------!--------!--------!
5710  ON Z2 GOTO     5719  ,  5743  ,  5734  ,  5764           : REM BRANCH
5713 REM **       !--------!--------!--------!--------!
5716 REM **
5719  GOSUB ' 191 (7,2)                                       : REM RCVRBL DAT ERR
5722  GOSUB ' 191 (8,9)                                       : REM TRANSMIT DAT ERR
5725  GOSUB ' 191 (8,10)                                      : REM DAT OVERFLOW ERR
5728  GOSUB ' 191 (9,10)                                      : REM DAT OVR/UNDR FLOW
5731 REM **
5734  GOSUB ' 191 (7,1)                                       : REM IRRCVRBL DAT ERR
5737  GOSUB ' 191 (7,4)                                       : REM WRITE DISABLED
5740 REM **
5743  GOSUB ' 191 (7,3)                                       : REM BEG OF TAPE
5746  GOSUB ' 191 (7,5)                                       : REM END OF TAPE
5749  GOSUB ' 191 (7,6)                                       : REM END OF FILE
5752  GOSUB ' 191 (7,7)                                       : REM DEV NOT READY
5755  GOSUB ' 191 (8,8)                                       : REM MICRO CMND ERR
5758  IF          Z3 = 3                        THEN 5692     : REM REPROCS RECORD
5761 REM **
5764  RETURN                                                  : REM EXIT
```

```
5767 REM **
5770 REM **
5773 REM **  Subroutine  ********************************************** 1 9 1
5776 REM **  Abstract >
5779 REM **    To mask the GIO register buffer with error
5782 REM **    mask buffer and display error messages if any.
5785 REM **  Variables >
5788 REM **    Z5 > supress non-fatal msgs   ! Z7 > buffer register
5791 REM **    Z8 > error index to check     ! Z4 > transfer bufr size
5794 REM **    Z9 > characters processed     ! Z1$() > micro cmnd buffer
5797 REM **    Z2$() > error mask buffer     ! Z3$ > non-fatal err msg
5800 REM **    Z3 > return error index       ! Z0 > multi volume flag
5803 REM **
5806 REM ***********************************************
5809  DEFFN ' 191 (Z7,Z8)                                  : REM ENTRY
5812  IF          Z3 <> 0                       THEN 5956 : REM PRIVIOUS ERR
5815  IF          Z7 <> 9                       THEN 5842 : REM NOT COUNT REG
5818  GOSUB ' 191 (7,6)                                  : REM EOF TEST
5821  IF          Z3 = 4                        THEN 5956 : REM EOF RECORD
5824  GOSUB ' 191 (7,5)                                  : REM EOT TEST
5827  IF          Z3 = 3                        THEN 5956 : REM EOT ERROR
5830              Z9 = FNZ(13)                            : REM CMPUT TRANSFR CNT
5833  IF          Z4 <> Z9                      THEN 5881 : REM OVR/UNDR FLOW ERR
5836  GOTO        5956                                    : REM BRANCH TO EXIT
5839 REM **
5842  IF          Z8 <> 5                       THEN 5854 : REM NOT EOT TEST
5845  IF          Z1 <> 9                       THEN 5854 : REM NOT REWIND
5848  IF          Z0 = 99                       THEN 5869 : REM PREVIOUS EOT ERR
5851 REM **
5854  AND         (Z2$(Z8),STR(Z1$(13),Z7,1))             : REM MASK REGISTER
5857  IF          Z2$(Z8) = HEX(00)             THEN 5956 : REM NO ERROR
5860 REM **
5863 REM **       ! 01 F ! 02 F ! 03 I ! 04 I ! 05 I  !
5866 REM **       !------!------!------!------!------!
5869  ON Z8 GOTO    5881 , 5881 , 5914 , 5917 , 5920  ,
                    5926 , 5881 , 5881 , 5881 , 5881  : REM BRANCH
5872 REM **       !------!------!------!------!------!
5875 REM **       ! 06 I ! 07 F ! 08 F ! 09 F ! 10 F  !
5878 REM **
5881  PRINT       HEX(01070C0C0C);                        : REM POSITION CUSOR
5884  PRINT       "Touch 'CONTINUE' 'RETURN' to Attempt Recovery";
5887 REM **
5890 REM **       ! 01 F ! 02 F ! 03 I ! 04 I ! 05 I  !
5893 REM **       !------!------!------!------!------!
5896  ON Z8 GOTO    5908 , 5911 , 5914 , 5917 , 5923  ,
                    5926 , 5929 , 5932 , 5935 , 5938  : REM BRANCH
5899 REM **       !------!------!------!------!------!
5902 REM **       ! 06 I ! 07 F ! 08 F ! 09 F ! 10 F  !
5905 REM **
```

```
5908  STOP        "IRRECOVERABLE DATA ERROR"     : Z=3 : GOTO 5956
5911  STOP        "RECOVERABLE DATA ERROR"       : Z=3 : GOTO 5956
5914  Z3 = 1      : Z3$ = "BEGINING OF TAPE"             : GOTO 5944
5917  Z3 = 2      : Z3$ = "TAPE WRITE DISABLED"          : GOTO 5944
5920  Z3 = 3      : Z3$ = "END OF TAPE"  : Z0 = 99       : GOTO 5881
5923  STOP        "MOUNT FIRST/NEXT TAPE VOLUME..."      : GOTO 5944
5926  Z3 = 4      : Z3$ = "END OF FILE"                  : GOTO 5944
5929  STOP        "TAPE UNIT NOT READY"          : Z=3 : GOTO 5956
5932  STOP        "INVALID MICRO COMMAND SEQUENCE"       : GOTO 5956
5935  STOP        "DATA READ ERROR"              : Z=3 : GOTO 5956
5938  STOP        "DATA OVER/UNDER FLOW ERROR"   : Z=3 : GOTO 5956
5941 REM **
5944  IF          Z5 = 0                    THEN 5956 : REM SUPPRESS MESG
5947  PRINT       HEX(01070C0C0C);                    : REM POSITION CUSOR
5950  PRINT       STR(Z3$,1,40);                      : REM PRINT MSG
5953 REM **
5956  RETURN                                          : REM EXIT
```

AUTHOR System: Phase 3 - ATH3LST4 - # MAP

```
5692  - 5758
5719  - 5710
5734  - 5710
5743  - 5710
5764  - 5710
5842  - 5815
5854  - 5842 5845
5869  - 5848
5881  - 5833 5869 5920
5908  - 5896
5911  - 5896
5914  - 5869 5896
5917  - 5869 5896
5920  - 5869
5923  - 5896
5926  - 5869 5896
5929  - 5896
5932  - 5896
5935  - 5896
5938  - 5896
5944  - 5914 5917 5923 5926
5956  - 5812 5821 5827 5836 5857 5908 5911 5929 5932 5935 5938 5944
```

```
Q2$(   - 5692
Y      - 5626
Z      - 5623 5908 5911 5929 5935 5938
Z0     - 5848 5920
Z1     - 5617 5692 5845
Z1$(   - 5620 5623 5626 5632 5635 5638 5641 5644 5647 5650 5653 5656 5659 5662
         5665 5668 5692 5854
Z2     - 5617 5710
Z2$(   - 5620 5674 5677 5680 5683 5686 5854 5857
Z3     - 5617 5698 5758 5812 5821 5827 5914 5917 5920 5926
Z3$    - 5620 5914 5917 5920 5926 5950
Z4     - 5617 5692 5833
Z5     - 5617 5944
Z6     - 5617
Z7     - 5809 5815 5854
Z8     - 5809 5842 5854 5857 5869 5896
Z9     - 5830 5833
```

```
AUTHOR System: Phase 3 - ATH3LST4 - ' MAP

0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
5617 DEFFN' 190
5809 DEFFN' 191
       - 5719 5722 5725 5728 5734 5737 5743 5746 5749 5752 5755 5818 5824
```

5.35.5 ATH3LST5

This part of the 'ATH3LSTP' program contains subroutines 201 to 211.

The remainder of this section presents the 'ATH3LST5' program listing and reference maps.

```
6000 %## Program > AUTHOR/3  (ATH3LST5)   Version > 2.0   2200 T/VP  780323
6004 REM **  Subroutine  ********************************************** 2 0 1
6008 REM **  Abstract >
6012 REM **    To select the output device onto which the data base
6016 REM **    will be output.
6020 REM **  Variables >
6024 REM **    W6 > output device indx  !  RO$() > input register
6028 REM **
6032 REM **********************************************
6036  DEFFN ' 201 (W6)                                    : REM ENTRY
6040  IF          W6 = 7                        THEN 6180 : REM RETRUN FROM EDIT
6044  IF          W6 = 8                        THEN 6180 : REM RETRUN FROM EDIT
6048  PRINT       HEX(03)                                 : REM CLR CRT SCREEN
6052  GOSUB ' 101 (10,1,0,0)                              : REM POSITION CUSOR
6056  SELECT      PRINT 005                               : REM SEL OUTPUT MODE
6060  PRINTUSING  6160, " "                               : REM PRINT MENU
6064  PRINTUSING  6164, " "                               : REM PRINT MENU
6068  PRINTUSING  6168, " "                               : REM PRINT MENU
6072  PRINTUSING  6172, " "                               : REM PRINT MENU
6076  SELECT      PRINT 405                               : REM SEL OUTPUT MODE
6080  GOSUB ' 101 (5,1,0,0)                               : REM POSITION CUSOR
6084  PRINTUSING  6176, " ";                              : REM PRINT PROMPT
6088              STR(R8$(9),5,1) = " "                   : REM CLR REGISTER
6092  KEYIN       RO$(1), 6108, 6108                      : REM INPUT RESPONCE
6096  $GIO WAIT   /005(R8$(1),R8$(9))                     : REM INPUT WAIT
6100  GOTO        6092                                    : REM CONT INPUT WAIT
6104 REM **
6108  IF          RO$(1) > HEX(39)              THEN 6116 : REM INVALID INPUT
6112  IF          RO$(1) > HEX(30)              THEN 6140 : REM VALID INPUT
6116  GOSUB ' 101 (5,50,0,0)                              : REM POSITIN CUSOR
6120  PRINT       "RE-ENTER"                              : REM PRINT ERR MSG
6124  $GIO TONE   /005(R8$(4),R8$(9))                     : REM PAUSE TONE
6128  GOSUB ' 101 (5,50,1,0)                              : REM ERASE ERR MSG
6132  GOTO        6092                                    : REM RE-ENTER
6136 REM **
6140  CONVERT     RO$(1) TO W6                            : REM STOR DEVICE INDX
6144  PRINT       W6                                      : REM DISPLAY VALUE
6148 REM **
6152 REM **                                                OUTPUT IMAGE FORMATS
6156 REM **
6160  % Output Selection     #    1 - CRT          <005>
6164  %                      #    2 - TTY          <011>
6168  %                      #    3 - LPT          <015>
6172  %                      #    1 - EDIT MODE    <005>
6176  % Enter Option From Output Selection Menu > #
6180  RETURN                                              : REM EXIT
```

```
6184 REM **
6188 REM **
6192 REM **  Subroutine  ********************************************** 2 0 2
6196 REM **  Abstract >
6200 REM **    To initialize text and index buffers before
6204 REM **    begining list processing.
6208 REM **  Variables >
6212 REM **    D9 > scratch addrs regstr  ! D1 > current indx sctr
6216 REM **    D2 > sub-sctr index        ! D6 > current output disk
6220 REM **    W2$() > output page buffer ! H1$() > graphic index table
6224 REM **    H2$() > graphic i/o buffer ! Q1$() > index sctr i/o bufr
6228 REM **    W6 > output device index   !
6232 REM **
6236 REM ***********************************************
6240  DEFFN ' 202                                          : REM ENTRY
6244  DIM          H1$(30)11,H2$(11)1                      : REM DEFINE WORK BUFRS
6248  IF           W6 = 5                        THEN 6424 : REM PRINT INSTRUCTNS
6252  IF           W6 = 8                        THEN 6528 : REM RETURN FROM EDIT
6256               H1$( 1)=HEX(0413222E3A080B0808035F)     : REM DEFN GRAPHIC CODE
6260               H1$( 2)=HEX(0314232F380D0B0808035F)     : REM DEFN GRAPHIC CODE
6264               H1$( 3)=HEX(0312232F380B0D0808035F)     : REM DEFN GRAPHIC CODE
6268               H1$( 4)=HEX(0312212F3808080A08035F)     : REM DEFN GRAPHIC CODE
6272               H1$( 5)=HEX(0312212D3808080B0A035F)     : REM DEFN GRAPHIC CODE
6276               H1$( 6)=HEX(3C033B0109030700005F21)     : REM DEFN GRAPHIC CODE
6280               H1$( 7)=HEX(3D033C0109030700005F21)     : REM DEFN GRAPHIC CODE
6284               H1$( 8)=HEX(3D033C0109030700005F21)     : REM DEFN GRAPHIC CODE
6288               H1$( 9)=HEX(3D033C0109030700005F21)     : REM DEFN GRAPHIC CODE
6292               H1$(10)=HEX(3D033C0109030700005F21)     : REM DEFN GRAPHIC CODE
6296               H1$(11)=HEX(030F121E01010101000021)     : REM DEFN GRAPHIC CODE
6300               H1$(12)=HEX(020F131F02020101000021)     : REM DEFN GRAPHIC CODE
6304               H1$(13)=HEX(020E111E01010202000021)     : REM DEFN GRAPHIC CODE
6308               H1$(14)=HEX(020E111D01010101000021)     : REM DEFN GRAPHIC CODE
6312               H1$(15)=HEX(020E111D01010101000021)     : REM DEFN GRAPHIC CODE
6316               H1$(16)=HEX(212A2D3601010101000021)     : REM DEFN GRAPHIC CODE
6320               H1$(17)=HEX(22282E3701010101000021)     : REM DEFN GRAPHIC CODE
6324               H1$(18)=HEX(22282E3701010101000021)     : REM DEFN GRAPHIC CODE
6328               H1$(19)=HEX(202A2E3702020101000021)     : REM DEFN GRAPHIC CODE
6332               H1$(20)=HEX(20292C3601010202000021)     : REM DEFN GRAPHIC CODE
6336               H1$(21)=HEX(1838380101070101095FA1)     : REM DEFN GRAPHIC CODE
6340               H1$(22)=HEX(183C3D0101030101A5FA1)      : REM DEFN GRAPHIC CODE
6344               H1$(23)=HEX(183C3D0101030101A5FA1)      : REM DEFN GRAPHIC CODE
6348               H1$(24)=HEX(183C3D0101030101A5FA1)      : REM DEFN GRAPHIC CODE
6352               H1$(25)=HEX(183C3D0101030101A5FA1)      : REM DEFN GRAPHIC CODE
6356               H1$(26)=HEX(101F2B3702020202050002D)    : REM DEFN GRAPHIC CODE
6360               H1$(27)=HEX(11202C3802020202050002D)    : REM DEFN GRAPHIC CODE
6364               H1$(28)=HEX(0F202C3802020202050002D)    : REM DEFN GRAPHIC CODE
6368               H1$(29)=HEX(0F1E2C3802020202050002D)    : REM DEFN GRAPHIC CODE
6372               H1$(30)=HEX(0F1E2A3802020202050002D)    : REM DEFN GRAPHIC CODE
```

```
6376  IF          W6 = 7                          THEN 6528 : REM RETURN FROM EDIT
6380  INIT(20)    Q2$()                                     : REM CLR BUFFR
6384  INIT(20)    Q1$()                                     : REM CLR BUFFR
6388  INIT(00)    H2$()                                     : REM CLR BUFFR
6392              D1 = 0                                    : REM INIT SCTR ADRS
6396              D2 = 128                                  : REM INIT SUB-SCTR
6400  DATA LOAD   BA T #3, (0,D9)Q1$()                      : REM LOAD ZERO SCTR
6404  ROTATE      (STR(Q1$(25),1,1),4)                      : REM CONV UPR BITS
6408              D6 = VAL(STR(Q1$(25),1,1))                : REM STOR VOL ID
6412  DATA LOAD   BA T #3, (1,D9)Q1$()                      : REM LOAD INDX SCTR
6416 REM **
6420  IF          W6 <> 1                         THEN 6528 : REM NOT CRT OUTPUT
6424  SELECT      PRINT 005                                 : REM SEL OUTPUT MODE
6428  PRINT       HEX(03)                                   : REM CLR CRT SCREEN
6432  RESTORE     1                                         : REM RESET DATA INDX
6436  FOR         W9 = 1 TO 13                              : REM BEG INST LOOP
6440  READ        R9$(1)                                    : REM READ INST LINE
6444  PRINTUSING  6468, R9$(1)                              : REM PRINT INST LINE
6448  NEXT        W9                                        : REM FETCH NEXT LINE
6452              W6 = 1                                    : REM RESET DEV INDX
6456 REM **
6460 REM **                                                  OUTPUT IMAGE FORMAT
6464 REM **
6468  % ############################################################
6472  DATA "Instructions for Referenceing pages:"
6476  DATA " - After each page is displayed a prompt will appear in"
6480  DATA "   the lower left corner of screen."
6484  DATA " - Once you have entered the next page number touch"
6488  DATA "   the 'RETURN' key."
6492  DATA " - To perform special functions you may enter:"
6496  DATA "     START - Begin displaying Learning Module."
6500  DATA "     NEXT  - Go to next sequintal page."
6504  DATA "     STOP  - Stop listing."
6508  DATA "     PRINT - Print current page on another device."
6512  DATA "     LIST  - List the entire learning module."
6516  DATA "     EDIT  - Edit current page."
6520  DATA "     HELP  - This text."
6524 REM **
6528  RETURN                                                : REM EXIT
```

```
6532 REM **
6536 REM **
6540 REM **  Subroutine  ********************************************** 2 0 3
6544 REM **  Abstract >
6548 REM **    To load first/next page into text buffer.  If CRT
6552 REM **    output wait for next address.
6556 REM **    Option - from CRT mode alow current buffer to
6560 REM **             be output to another device.
6564 REM **  Variables >
6568 REM **    W6 > output device index    ! W3 > temp change of dev flg
6572 REM **    D7$ > input pag adres       !
6576 REM **
6580 REM ************************************************
6584  DEFFN ' 203                                          : REM ENTRY
6588  IF          W6 = 7                        THEN 6784  : REM RETURN FORM EDIT
6592  IF          W6 = 8                        THEN 6692  : REM RETURN FORM EDIT
6596  GOSUB ' 205 (W6,W3,D7$)                              : REM FETCH NXT ADRS
6600  IF          D7$ <> "STOP"                 THEN 6652  : REM NOT END OF FILE
6604  IF          W4$ <> "TAPE"                 THEN 6620  : REM NO SPCL SCANS
6608  SELECT #5   215                                      : REM SEL LPT DEV
6612  SELECT TAPE 078                                      : REM SEL TAPE DEV
6616  GOSUB ' 108 (4)                                      : REM PRINT OVRLAYS
6620  PRINT       HEX(03)                                  : REM CLR CRT SCREEN
6624  SELECT      PRINT 005                                : REM SEL OUTPUT MODE
6628  PRINTUSING  6852, " "                                : REM PRINT HEADING
6632  PRINTUSING  6856, " "                                : REM PRINT HEADING
6636  PRINTUSING  6860, " "                                : REM PRINT HEADING
6640  GOSUB ' 101 (12,1,0,0)                               : REM POSITION CUSOR
6644  END                                                  : REM END EXECUTION
6648 REM **
6652  IF          D7$ <> "PRIN"                 THEN 6720  : REM NOT LIST CMND
6656  GOSUB ' 101 (14,1,3,30)                              : REM POSITION CUSOR
6660  SELECT      PRINT 005                                : REM SEL OUTPUT MODE
6664  PRINTUSING  6864, " "                                : REM PRINT MENU
6668  PRINTUSING  6868, " "                                : REM PRINT MENY
6672  PRINTUSING  6872, " ";                               : REM PRINT PROMPT
6676  SELECT      PRINT 405                                : REM SEL OUTPUT MODE
6680              W6 = 2                                   : REM SET DEFAULT
6684  KEYIN       R0$(1), 6688, 6688       : GOTO 6684     : REM RESPONCE WAIT
6688  IF          R0$(1) <> "3"                 THEN 6696  : REM SET FLAG
6692              W6 = 3                                   : REM SET DEV INDX
6696              W3 = 99                                  : REM SET TMP DEV FLG
6700  GOSUB ' 101 (16,1,1,30)                              : REM POSITION CUSOR
6704  PRINT       "System Working..."                      : REM PRINT SYS WAIT
6708              D7$ = Q1$(D2+1)                          : REM SET PAG REFRNCE
6712  GOTO        6836                                     : REM LOAD TEXT BUFR
6716 REM **
6720  IF          D7$ <> "ERRO"                 THEN 6748  : REM NO ERR DETCTED
6724  GOSUB ' 101 (15,9,0,0)                               : REM POSITION CUSOR
6728  PRINT       "RE-ENTER"                               : REM PRINT ERR MSG
6732  $GIO TONE   /005(R8$(4),R8$(9))                      : REM PAUSE TONE
6736  GOSUB ' 101 (15,9,1,10)                              : REM ERASE ERR MSG
6740  GOTO        6596                                     : REM RE-ENTER ADRS
```

```
6744 REM **
6748  IF          D7$ <> "HELP"              THEN 6768 : REM NOT HELP CMND
6752              W6  =  5                             : REM SET FLAG
6756  GOSUB ' 202                                      : REM DISPLAY INST
6760  GOTO        6596                                 : REM RE-ENTER ADRS
6764 REM **
6768  IF          D7$ <> "EDIT"              THEN 6800 : REM NOT EDIT CMND
6772              W6  =  7                             : REM SET RETURN INDX
6776  LOAD DC T    #1, "ATH3EDTP" 1500,9999
6780  REM **
6784              W6  =  1                             : REM RESET DEV INDX
6788              D7$ =  Q1$(D2+1)                     : REM SET PAG REFRENCE
6792  GOTO        6836                                 : REM LOAD TEXT BUFFR
6796 REM **
6800  IF          D7$ <> "LIST"              THEN 6836 : REM NOT LIST CMND
6804              W6  =  0                             : REM RESET DEV INDX
6808              W2$ =  " "   :   W3$ =  " "          : REM CLR DISPLAY FLGS
6812              W4$ =  " "   :   W5$ =  " "          : REM CLR DISPLAY FLGS
6816              W6$ =  " "                           : REM CLR DISPLAY FLGS
6820  GOSUB ' 201 (W6)                                 : REM SEL OUTPUT DEV
6824  GOSUB ' 202                                      : REM INITIALIZE
6828  GOTO         6584                                : REM LOAD FIRST PAGE
6832 REM **
6836  GOSUB ' 207 (D7$)                                : REM LOAD TEXT BUFFER
6840 REM **
6844 REM **                                              OUTPUT IMAGE FORMAT
6848 REM **
6852  % AUTHOR;  An Automatic Authoring System #
6856  % Phase III   Construction Processing    #
6860  % L I S T I N G    C O M P L E T E D     #
6864  % Menu #  2 - TTY <011>,
6868  %      #  3 - LPT <015>
6872  % Enter Temp Device > #
6876  RETURN                                           : REM EXIT
```

```
6880 REM **
6884 REM **
6888 REM **  Subroutine  ********************************************** 2 0 4
6892 REM **  Abstract >
6896 REM **    To scan buffer for graphic keyword.
6900 REM **    If found process graphic function.
6904 REM **  Variables >
6908 REM **    TO > linear key position  ! T1 > graphic code indx
6912 REM **    T2 > row indx of graphic  ! R2$() > return srch bufr
6916 REM **    W2$() > output page bufr  ! R5$() > single char storage
6920 REM **    RO$() > input register    ! D4$ > search id
6924 REM **
6928 REM ************************************************
6932  DEFFN ' 204                                          : REM ENTRY
6936  IF          W6 = 4                        THEN 6996 : REM END LPT TAP FIL
6940  IF          W6 = 1                        THEN 6996 : REM CRT OUTPUT FIL
6944              D4$ = ":dia"                            : REM SET SRCH ID
6948  MAT SEARCH  Q2$(), = D4$ TO R2$()                   : REM SRCH FOR KEY
6952  IF          R2$(1) = HEX(0000)            THEN 6996 : REM NO KEYWORDS
6956              TO = FNR(VAL(R2$(1)))                   : REM CMPUT POSITION
6960  MAT COPY    Q2$()<TO+4,1> TO RO$()<1,1>             : REM FETCH GRPHC CODE
6964  MAT COPY    R5$()<1,1> TO Q2$()<TO,5>               : REM CLR KEYWORD
6968  CONVERT     RO$(1) TO T1                            : REM CONV GRPHC CODE
6972              T1 = T1 + 1                             : REM INCR GRPHC INDX
6976              T2 = INT((TO-1)/64) + 1                 : REM CMPUT ROW INDX
6980  IF          T1 > 5                        THEN 6996 : REM INVALID KEY
6984  IF          T1 < 1                        THEN 6996 : REM INVALID KEY
6988  GOSUB ' 208 (T2,T2+1,T1)                            : REM PROCES GRAPHIC
6992 REM **
6996  RETURN                                              : REM EXIT
```

```
7000 REM **
7004 REM **
7008 REM **  Subroutine  ********************************************** 2 0 5
7012 REM **  Abstract >
7016 REM **    To compute the next frame address. If CRT output
7020 REM **    input frame address.
7024 REM **  Variables >
7028 REM **    W6 > output device index      ! Q1$() > index sctr buffer
7032 REM **    W3 > temp dev change flag     ! D7$ > frame address register
7036 REM **    D1 > index sctr address       ! D2 > sub-sctr index
7040 REM **
7044 REM ************************************************
7048  DEFFN ' 205 (W6,W3,D7$)                              : REM ENTRY
7052  IF          W3 <> 99                       THEN 7076 : REM NOT DEV CHANGE
7056              W3 = 00                                  : REM RESET FLAG
7060              W6 = 01                                  : REM RESET DEVICE INDX
7064              D7$ = Q1$(D2+1)                          : REM SET BEG ADDRESS
7068  GOTO        7224                                     : REM BRANCH TO EXIT
7072 REM **
7076  IF          W6 = 01                        THEN 7136 : REM INPUT FRAME ADRS
7080              D2 = D2 + 2                              : REM INCR SUBSCTR INDX
7084  IF          D2 < 128                       THEN 7116 : REM WITHN LIMITS
7088              D1 = D1 + 1                              : REM INCR SCTR INDX
7092  IF          D1 < 17                        THEN 7104 : REM LOAD INDX SCTR
7096  GOSUB ' 000                                          : REM SYSTEM ERROR
7100 REM **
7104  DATA LOAD   BA T #3, (D1,D9) Q1$()                   : REM LOAD INDX SCTR
7108              D2 = 1                                   : REM RESET SUBSCTR INDX
7112 REM **
7116              D7$ = Q1$(D2+1)                          : REM STOR LOCATION
7120  IF          Q1$(D2) <> HEX(FFFF)           THEN 7224 : REM NOT END OF FILE
7124              D7$ = "STOP"                             : REM SET EOF FLAG
7128  GOTO        7224                                     : REM BRANCH TO EXIT
7132 REM **
7136  GOSUB ' 101 (15,1,2,20)                              : REM POSITION CUSOR
7140  PRINT       HEX(20203F3F3F3F080808080808);           : REM BACKGROUND
7144              D7$ = "NEXT"                             : REM SET DEFALUT CMND
7148  INPUT       D7$                                      : REM INPUT FRAME ADRS
7152  PRINT       "System Working...";                     : REM PRINT SYS BUSY
7156  AND         (D7$,DF)                                 : REM FORCE UPPERCASE
7160  $TRAN       (D7$,R4$()<193,24>) R                    : REM CONV DIGITS
7164 REM **
7168  IF          D7$ = "STOP"                   THEN 7224 : REM VALID INPUT
7172  IF          D7$ = "LIST"                   THEN 7224 : REM VALID INPUT
7176  IF          D7$ = "HELP"                   THEN 7224 : REM VALID INPUT
7180  IF          D7$ = "EDIT"                   THEN 7224 : REM VALID INPUT
7184  IF          D7$ = "PRIN"                   THEN 7224 : REM VALID INPUT
7188  IF          D7$ = "NEXT"                   THEN 7080 : REM INCR TO NEXT
7192  IF          D7$ <> "STAR"                  THEN 7212 : REM NOT KEY COMMAND
7196              D1  = 0                                  : REM RESET SCTR ADRS
7200              D2  = 129                                : REM RESET SUBSCTR INDX
7204  GOTO        7080                                     : REM INCR TO NEXT
7208 REM **
7212  GOSUB ' 209 (D7$)                                    : REM CONV TO BIN ADRS
7216  GOSUB ' 206 (D7$,D1,D2)                              : REM FETCH LOCATION
7220 REM **
7224  RETURN                                               : REM EXIT
```

```
7228 REM **
7232 REM **
7236 REM **  Subroutine ********************************************* 206
7240 REM **  Abstract >
7244 REM **    To perform and index lookup and compute the location
7248 REM **    of next page.
7252 REM **  Variables >
7256 REM **    D7$ > binary page address      ! D3 > temp indx sctr adrs
7260 REM **    D4 > temp subsctr index        ! D1 > indx sctr adrs
7264 REM **    D2 > sub-sctr index            ! Q1$() > indx sctr buffer
7268 REM **    R2$() > return search buffer   ! D5 > temp loop index
7272 REM **    D9 > scatch addres register    ! D6 > current output disk
7276 REM **
7280 REM ***********************************************
7284  DEFFN ' 206 (D7$,D1,D2)                              : REM ENTRY
7288  IF          D7$ = "ERRO"                THEN 7384  : REM INVALID ADDRESS
7292              D3 = D1                                : REM SAVE SCTR ADRS
7296              D4 = D2                                : REM SAVE SCTR INDX
7300  MAT SEARCH  Q1$(),=STR(D7$,1,2) TO R2$() STEP 4    : REM SRCH CRNT SCTR
7304  IF          R2$(1) <> HEX(0000)         THEN 7368  : REM ID FOUND
7308              D3 = 0                                 : REM SET SCTR ADRS
7312 REM **
7316  FOR         D5 = 1 TO 10                           : REM BEG INDX LOOKUP
7320  DATA LOAD   BA T #3, (D5,D9) Q1$()                 : REM LOAD INDX SCTR
7324  MAT SEARCH  Q1$(),=STR(D7$,1,2) TO R2$() STEP 4    : REM SRCH INDX SCTR
7328  IF          R2$(1) = HEX(0000)          THEN 7340  : REM ID NOT FOUND
7332              D3 = D5                                : REM SAVE NEW SCTR ADRS
7336              D5 = 10                                : REM SET END OF LOOKUP
7340  NEXT        D5                                     : REM CONT INDX LOOKUP
7344 REM **
7348  IF          D3 <> 0                     THEN 7368  : REM NEW INDX SCTR
7352              D7$ = "ERRO"                           : REM ID NOT FOUND
7356  DATA LOAD   BA T #3, (D1,D9) Q1$()                 : REM RELOAD INDX
7360  GOTO        7384                                   : REM BRANCH TO EXIT
7364 REM **
7368              D1 = D3                                : REM UPDATE SCTR ADRS
7372              D2 = INT(FNR(VAL(R2$(1)))/2) + 1       : REM CMPUT SCTR INDEX
7376              D7$ = Q1$(D2+1)                        : REM STOR PAC REFERNCE
7380 REM **
7384  RETURN                                             : REM EXIT
```

```
7388 REM **
7392 REM **
7396 REM **  Subroutine  ********************************************** 2 0 7
7400 REM **  Abstract >
7404 REM **    To load text buffer from disk.
7408 REM **  Variables >
7412 REM **    D7$ > stating location on disk ! W3 > temp device chang flg
7416 REM **                                   ! Q2$() > text buffer (32)
7420 REM **    D9 > temp loop index           ! W6 > output device indx
7424 REM **    W8 > text bufr scale factor    ! Q9$() > sector load buffer
7428 REM **
7432 REM ***********************************************
7436  DEFFN ' 207 (D7$)                                    : REM ENTRY
7440  IF          W3 = 99                       THEN 7448 : REM EXPAND BUFFER
7444  INIT(20)    Q2$()                                   : REM CLR TEXT BUFR
7448  GOSUB ' 211 (D7$)                                   : REM CHK DSK VOL ID
7452 REM **
7456  FOR         W9 = 1 TO 1024 STEP 256                 : REM BEG DISK LOAD
7460  DATA LOAD   BA T #3, (D7$,D7$) Q9$()                : REM LOAD INPUT BUFR
7464  FOR         W8 = 1 TO 256  STEP 64                  : REM BEG BUFR LOAD
7468              W7 = (W8+(W9-1))*2 - 1                  : REM CMPUT SCALE FACTR
7472  MAT COPY    Q9$()<W8,64> TO Q2$()<W7,128>           : REM LOAD TEXT BUFR
7476  NEXT        W8                                      : REM CONT BUFR LOAD
7480  NEXT        W9                                      : REM CONT DISK LOAD
7484 REM **
7488  IF          W6 <> 1                       THEN 7516 : REM CRT OUTPUT DEV
7492 REM **
7496  FOR         W9 = 1 TO 2048 STEP 128                 : REM BEG BUFR CONTRACT
7500              W8 = INT((W9+1)/2)                      : REM COMPUT SCALE
7504  MAT COPY    Q2$()<W9,128> TO Q2$()<W8,64>           : REM LOAD TEXT BUFR
7508  NEXT        W9                                      : REM CONT BUFR LOAD
7512 REM **
7516  RETURN                                              : REM EXIT
```

```
7520 REM **
7524 REM **
7528 REM **  Subroutine  ********************************************** 2 0 8
7532 REM **  Abstract >
7536 REM **    To process special graphic functions.
7540 REM **  Variables >
7544 REM **    H1 > begining row of graphic  ! H2 > second row of graphic
7548 REM **    H3 > graphic function indx    ! H2$() > data i/o registers
7552 REM **    H1$() > data storage buffer   ! R9$() > temp data bufr
7556 REM **    H2$ > data unpack format      ! W8 > temp loop index
7560 REM **    W9 > temp loop index          !
7564 REM **
7568 REM ***********************************************
7572  DEFFN ' 208 (H1,H2,H3)                                  : REM ENTRY
7576  DIM         H2$(11)1,H2$2                               : REM DEFINE WORK BUFRS
7580  DEFFN       H(H) = VAL(H2$(H))                          : REM FUNCTION    H()
7584  DEFFN       I(I) = FNH(W9+I)                            : REM FUNCTION    I()
7588  DEFFN       J(J) = H1 + FNH(J)                          : REM FUNCTION    J()
7592              H2$  = HEX(A001)                            : REM DATA FORMAT
7596 REM **
7600  $UNPACK     (F=H2$) H1$(H3) TO H2$()                    : REM UNPACK GRAPHIC DAT
7604  INIT        (H2$(11)) R9$()                             : REM INIT CHAR BUFFER
7608  FOR         W9 = 1 TO 5                                 : REM BEG LOOP
7612  GOSUB ' 100 (H2,FNI(0),FNI(5),R9$(1),1)                 : REM LOAD TEXT BUFR
7616  NEXT        W9                                          : REM CONT LOOP
7620 REM **
7624  $UNPACK     (F=H2$) H1$(H3+5) TO H2$()                  : REM UNPACK GRAPHIC DAT
7628  INIT        (H2$(10)) R9$()                             : REM INIT CHAR BUFFER
7632  GOSUB ' 100 (FNJ(5),FNH(1),FNH(2),R9$(1),1)             : REM LOAD TEXT BUFR
7636  INIT        (H2$(11)) R9$()                             : REM INIT CHAR BUFFER
7640  GOSUB ' 100 (FNJ(5),FNH(3),FNH(4),R9$(1),1)             : REM LOAD TEXT BUFR
7644 REM **
7648  FOR         W8 = FNJ(6) TO FNJ(7) STEP 2                : REM BEG LOOP  #1
7652  FOR         W7 = H3+10 TO H3+15   STEP 5                : REM BEG LOOP  #2
7656  $UNPACK     (F=H2$) H1$(W7) TO H2$()                    : REM UNPACK GRAPHIC DAT
7660  INIT        (H2$(11)) R9$()                             : REM INIT CHAR BUFFER
7664  FOR         W9 = 1 TO 4                                 : REM BEG LOOP  #3
7668  GOSUB ' 100 (W8,FNI(0),FNI(4),R9$(1),1)                 : REM LOAD TEXT BUFFER
7672  NEXT        W9                                          : REM CONT LOOP #3
7676  NEXT        W7                                          : REM CONT LOOP #2
7680  NEXT        W8                                          : REM CONT LOOP #1
7684 REM **
7688  $UNPACK     (F=H2$) H1$(H3+20) TO H2$()                 : REM UNPACK GRAPHIC DAT
7692  INIT        (H2$(11)) R9$()                             : REM INIT CHAR BUFFER
7696  GOSUB ' 100 (FNJ(7),FNH(1),FNH(4),R9$(1),1)             : REM LOAD TEXT BUFFER
7700  GOSUB ' 100 (FNJ(8),FNH(2),FNH(5),R9$(1),1)             : REM LOAD TEXT BUFFER
7704  INIT        (H2$(10)) R9$()                             : REM INIT CHAR BUFFER
7708              STR(R9$(1),1,3) = "Yes"                     : REM INIT CHAR BUFFER
7712  GOSUB ' 100 (FNJ(9),FNH(3),FNH(6),R9$(1),1)             : REM LOAD TEXT BUFFER
7716 REM **
7720  $UNPACK     (F=H2$) H1$(H3+25) TO H2$()                 : REM UNPACK GRAPHIC DAT
7724  INIT        (H2$(11)) R9$()                             : REM INIT CHAR BUFFER
7728  FOR         W9 = 1 TO 4                                 : REM BEG LOOP
7732  GOSUB ' 100 (FNJ(9),FNI(0),FNI(4),R9$(1),1)             : REM LOAD TEXT BUFFER
7736  NEXT        W9                                          : REM CONT LOOP
7740 REM **
7744  RETURN                                                  : REM EXIT
```

```
7748 REM **
7752 REM **
7756 REM **  Subroutine  ********************************************** 2 0 9
7760 REM **  Abstract >
7764 REM **    To compute binary page address from physical page
7768 REM **    address.
7772 REM **  Variables >
7776 REM **    D7$ > page address           ! D8 > max rows  (linear format)
7780 REM **    W1 > fiche page number       ! D9 > max cols  (linear format)
7784 REM **    W2 > fiche row number        ! W3 > fiche col number
7788 REM **    R4$() > conversion buffer    ! W1$ > binary page address
7792 REM **
7796 REM ***********************************************
7800  DEFFN ' 209 (D7$)                                    : REM ENTRY
7804              D8 = 7                                   : REM SET LINEAR ROW IDX
7808              D9 = 14                                  : REM SET LINEAR COL IDX
7812  IF          STR(D7$,2,1) > HEX(40)     THEN 7856     : REM NUM IN FICH FORMAT
7816  IF          NUM(D7$) = 4               THEN 7832     : REM VALID NUM FORMAT
7820              D7$ = "ERRO"                             : REM INVALID INPUT
7824  GOTO        7912                                     : REM BRANCH TO EXIT
7828 REM **
7832  CONVERT     D7$ TO D7                                : REM CONV TO DEC
7836              W1=INT(D7/(D8*D9))+1                     : REM CMPUT PAGE NUMBR
7840              W2=INT(((D7-(W1-1)*(D8*D9))-1)/D9)+1     : REM CMPUT ROW NUMBR
7844              W3=D7-((W1-1)*(D8*D9)+(W2-1)*D9)         : REM CMPUT COL NUMBR
7848  GOTO        7904                                     : REM CMPUT BIN ADDRS
7852 REM **
7856  $TRAN       (D7$<2,1>,R4$()<225,18>) OF              : REM CONV ROW INDEX
7860  IF          NUM(STR(D7$,1,1)) <> 1     THEN 7876     : REM INVALID INPUT
7864  IF          NUM(STR(D7$,3,2)) <> 2     THEN 7876     : REM INVALID INPUT
7868  IF          STR(D7$,2,1) > HEX(0F)     THEN 7876     : REM INVALID INPUT
7872  IF          STR(D7$,2,1) > HEX(00)     THEN 7888     : REM VALID INPUT
7876              D7$ = "ERRO"                             : REM INVALID INPUT
7880  GOTO        7912                                     : REM BRANCH TO EXIT
7884 REM **
7888  CONVERT     STR(D7$,1,1) TO W1                       : REM CMPUT PAGE NUMBR
7892  CONVERT     STR(D7$,3,2) TO W3                       : REM CMPUT COL NUMBR
7896              W2 = VAL(STR(D7$,2,1))                   : REM CMPUT ROW NUMBR
7900 REM **
7904  GOSUB ' 103 (W1$,W1,W2,W3,0,1)                       : REM CONV TO BINARY
7908              D7$ = W1$                                : REM SAVE BINARY ADDRS
7912  RETURN                                               : REM EXIT
```

```
7916 REM **
7920 REM **
7924 REM **  Subroutine  ********************************************** 2 1 0
7928 REM **  Abstract >
7932 REM **    To print text buffer onto the selected output device.
7936 REM **  Variables >
7940 REM **    W6 > output device indx      ! W3 > temp change of dev flag
7944 REM **    W4$ > device option flag     !
7948 REM **
7952 REM ***********************************************
7956  DEFFN ' 210 (W6)                                    : REM ENTRY
7960  GOSUB ' 108 (W6)                                    : REM OUTPUT BUFFER
7964  RETURN                                              : REM EXIT
```

```
7968 REM **
7972 REM **
7976 REM **  Subroutine  ********************************************** 2 1 1
7980 REM **  Abstract >
7984 REM **    To check that right output disk is mounted before
7988 REM **    sector lookup.
7992 REM **  Variables >
7996 REM **    Q9$() > scratch i/o buffer    ! D7$ > page location addres
8000 REM **    D6 > current output disk      ! W9 > temp volume id
8004 REM **    R0$() > input register        ! D9 > scratch adrs register
8008 REM **
8012 REM ************************************************
8016  DEFFN ' 211 (D7$)                                    : REM ENTRY
8020  MAT REDIM   Q9$(64)4                                 : REM DEFINE WORK BUFR
8024              STR(D7$,3,2) = STR(D7$,1,2)              : REM SAVE PAG LOCATION
8028  AND         (STR(D7$,1,1),0F)                        : REM MASK OUT VOL ID
8032  AND         (STR(D7$,3,1),F0)                        : REM MASK OUT LOCATION
8036  ROTATE      (STR(D7$,3,1),4)                         : REM CONV UPR BITS
8040              W9 = VAL(STR(D7$,3,1))                   : REM STOR TEMP VOL ID
8044 REM **
8048  IF          W9 = D6                       THEN 8112  : REM VOL IDS MATCH
8052  GOSUB ' 101 (15,1,2,0)                               : REM POSITION CUSOR
8056  PRINTUSING  8104, W9                                 : REM PRINT PROMPT
8060  KEYIN       R0$(1), 8076, 8076                       : REM RESPONCE WAIT
8064  $GIO PAUSE  /005(R8$(4),R8$(9))                      : REM PAUSE TONE
8068  GOTO        8060                                     : REM CONT WAIT
8072 REM **
8076  DATA LOAD   BA T #3, (0,D9) Q9$()                    : REM LOAD ZERO SCTR
8080  ROTATE      (STR(Q9$(13),1,1),4)                     : REM CONV UPR BITS
8084              D6 = VAL(STR(Q9$(13),1,1))               : REM STORE VOL ID
8088  GOTO        8048                                     : REM RE-CHECK IDS
8092 REM **
8096 REM **                                                  OUTPUT IMAGE FORMAT
8100 REM **
8104  % MOUNT Volume > ## Output Disk ...  'RETURN' to Continue
8108 REM **
8112  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH3LST5 - # MAP

```
(1500) - 6776
 6092  - 6100 6132
 6108  - 6092
 6116  - 6108
 6140  - 6112
 6160  - 6060
 6164  - 6064
 6168  - 6068
 6172  - 6072
 6176  - 6084
 6180  - 6040 6044
 6424  - 6248
 6468  - 6444
 6528  - 6252 6376 6420
 6584  - 6828
 6596  - 6740 6760
 6620  - 6604
 6652  - 6600
 6684  - 6684
 6688  - 6684
 6692  - 6592
 6696  - 6688
 6720  - 6652
 6748  - 6720
 6768  - 6748
 6784  - 6588
 6800  - 6768
 6836  - 6712 6792 6800
 6852  - 6628
 6856  - 6632
 6860  - 6636
 6864  - 6664
 6868  - 6668
 6872  - 6672
 6996  - 6936 6940 6952 6980 6984
 7076  - 7052
 7080  - 7188 7204
 7104  - 7092
 7116  - 7084
 7136  - 7076
 7212  - 7192
 7224  - 7068 7120 7128 7168 7172 7176 7180 7184
 7340  - 7328
 7368  - 7304 7348
 7384  - 7288 7360
 7448  - 7440
 7516  - 7488
 7832  - 7816
 7856  - 7812
 7876  - 7860 7864 7868
 7888  - 7872
 7904  - 7848
 7912  - 7824 7880
 8048  - 8088
 8060  - 8068
 8076  - 8060
 8104  - 8056
 8112  - 8048
(9999) - 6776
```

```
D1     - 6392 7088 7092 7104 7196 7216 7284 7292 7356 7368
D2     - 6396 6708 6788 7064 7080 7084 7108 7116 7120 7200 7216 7284 7296 7372
         7376
D3     - 7292 7308 7332 7348 7368
D4     - 7296
D4$    - 6944 6948
D5     - 7316 7320 7332 7336 7340
D6     - 6408 8048 8084
D7     - 7832 7836 7840 7844
D7$    - 6596 6600 6652 6708 6720 6748 6768 6788 6800 6836 7048 7064 7116 7124
         7144 7148 7156 7160 7168 7172 7176 7180 7184 7188 7192 7212 7216 7284
         7288 7300 7324 7352 7376 7436 7448 7460 7800 7812 7816 7820 7832 7856
         7860 7864 7868 7872 7876 7888 7892 7896 7908 8016 8024 8028 8032 8036
         8040
D8     - 7804 7836 7840 7844
D9     - 6400 6412 7104 7320 7356 7808 7836 7840 7844 8076
H      - 7580
H1     - 7572 7588
H1$(   - 6244 6256 6260 6264 6268 6272 6276 6280 6284 6288 6292 6296 6300 6304
         6308 6312 6316 6320 6324 6328 6332 6336 6340 6344 6348 6352 6356 6360
         6364 6368 6372 7600 7624 7656 7688 7720
H2     - 7572 7612
H2$    - 7576 7592 7600 7624 7656 7688 7720
H2$(   - 6244 6388 7576 7580 7600 7604 7624 7628 7636 7656 7660 7688 7692 7704
         7720 7724
H3     - 7572 7600 7624 7652 7688 7720
I      - 7584
J      - 7588
Q1$(   - 6384 6400 6404 6408 6412 6708 6788 7064 7104 7116 7120 7300 7320 7324
         7356 7376
Q2$(   - 6380 6948 6960 6964 7444 7472 7504
Q9$(   - 7460 7472 8020 8076 8080 8084
R0$(   - 6092 6108 6112 6140 6684 6688 6960 6968 8060
R2$(   - 6948 6952 6956 7300 7304 7324 7328 7372
R4$(   - 7160 7856
R5$(   - 6964
R8$(   - 6088 6096 6124 6732 8064
R9$(   - 6440 6444 7604 7612 7628 7632 7636 7640 7660 7668 7692 7696 7700 7704
         7708 7712 7724 7732
T0     - 6956 6960 6964 6976
T1     - 6968 6972 6980 6984 6988
T2     - 6976 6988
W1     - 7836 7840 7844 7888 7904
W1$    - 7904 7908
W2     - 7840 7844 7896 7904
W2$    - 6808
W3     - 6596 6696 7048 7052 7056 7440 7844 7892 7904
W3$    - 6808
W4$    - 6604 6812
W5$    - 6812
W6     - 6036 6040 6044 6140 6144 6248 6252 6376 6420 6452 6588 6592 6596 6680
         6692 6752 6772 6784 6804 6820 6936 6940 7048 7060 7076 7488 7956 7960
W6$    - 6816
W7     - 7468 7472 7652 7656 7676
W8     - 7464 7468 7472 7476 7500 7504 7648 7668 7680
W9     - 6436 6448 7456 7468 7480 7496 7500 7504 7508 7584 7608 7616 7664 7672
         7728 7736 8040 8048 8056
```

```
AUTHOR System: Phase 3 - ATH3LST5 - ' MAP

0000 DEFFN' 0
       - 7096
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 100
       - 7612 7632 7640 7668 7696 7700 7712 7732
???? DEFFN' 101
       - 6052 6080 6116 6128 6640 6656 6700 6724 6736 7136 8052
???? DEFFN' 103
       - 7904
???? DEFFN' 108
       - 6616 7960
6036 DEFFN' 201
       - 6820
6240 DEFFN' 202
       - 6756 6824
6584 DEFFN' 203
6932 DEFFN' 204
7048 DEFFN' 205
       - 6596
7284 DEFFN' 206
       - 7216
7436 DEFFN' 207
       - 6836
7572 DEFFN' 208
       - 6988
7800 DEFFN' 209
       - 7212
7956 DEFFN' 210
8016 DEFFN' 211
       - 7448
```

5.36 ATH3EDTP

The purpose of the 'ATH3EDTP' program is to allow the current page of the learning module to be edited. The 'ATH3EDTP' program is called from 'ATH3LSTP'.

The remainder of this section presents the 'ATH3EDTP' program listing and reference maps.

```
1500  %## Program > AUTHOR/3  (ATH3EDTP)  Version > 2.0  2200 T/VP  780323
1504  REM **  Abstract  ************************************************************
1508  REM **                                                                        **
1512  REM **    This routine will alow the current page buffer to be                **
1516  REM **    edited in a fashion similar to that of editing a item               **
1520  REM **    of the Phase II data base.                                          **
1524  REM **                                                                        **
1528  REM ************************************************************************
1532  REM **
1536  GOSUB ' 107                                          : REM INITIALIZATION
1540  GOSUB ' 108 (RO$(1))                                 : REM ENTER A CHARACTER
1544  GOSUB ' 109 (RO$(1))                                 : REM PROCESS CHARACTER
1548  IF          RO$(1) <> HEX(FF)          THEN 1540     : REM NOT END OF DATA
1552  RETURN                                               : REM RETURN TO LIST
```

```
1556 REM **
1560 REM **
1564 REM **  Subroutine  ********************************************** 1 0 7
1568 REM **  Function >  To Initialize pointers and buffer
1572 REM **  Variables >
1576 REM **    R1    > crt pos <row>        ! R2    > crt pos <col>
1580 REM **    R3    > linear bufr indx     ! R6    > top lin on crt
1584 REM **    R4    > bufr pos <row>       ! R5    > bufr pos <col>
1588 REM **    R1$()> storage buffer        ! R0$()> charactr bufr
1592 REM **    R9$()> scratch               ! R9    > scratch
1596 REM **    R5$()> search id's           ! R8$()> micro cmnds
1600 REM **    E1 > pre-existing data flg !
1604 REM **
1608 REM ***********************************************
1612  DEFFN ' 107                                          : REM ENTRY
1616  DIM          R1$(16,64)1                             : REM DEFINE EDIT BUFFER
1620               R1 = 1                                  : REM INIT CRT ROW POS
1624               R2 = 1                                  : REM INIT CRT COL POS
1628               R3 = 1                                  : REM INIT LINR BUF POS
1632               R4 = 1                                  : REM INIT BUFR ROW POS
1636               R5 = 1                                  : REM INIT BUFR COL POS
1640               R6 = 1                                  : REM INIT TOP OF SCREN
1644               E1 = 1                                  : REM CLR EXSTNG DAT FLG
1648  INIT(20)     R1$                                     : REM CLR CUSR RPEAT FLG
1652  INIT(20)     R2$                                     : REM CLR NUMBER BUFR
1656  INIT(20)     R1$()                                   : REM CLR TEXT BUFR
1660  INIT(20)     R9$()                                   : REM CLR SCRATCH BUFR
1664  INIT(20)     R0$()                                   : REM CLR ENTRY BUFR
1668  KEYIN        R0$(1), 1668, 1668                      : REM CLR ENTRY QUEUE
1672  MAT COPY     Q2$()<1025,1024> TO R1$()<1,1024>       : REM LOAD EDIT BUFFER
1676  $TRAN        (R1$()<1,1024>, R4$()<177,4>) R         : REM CONV 00 TO 20
1680  GOSUB ' 121 (R6)                                     : REM DISPLAY BUFFER
1684 REM **
1688  GOSUB ' 101 (16,1,1,7)                               : REM POSITION CUSOR
1692  PRINT        "Entry ";
1696  GOSUB ' 101 (R1,R2,0,0)                              : REM POSITION CUSUR
1700  RETURN                                               : REM EXIT
```

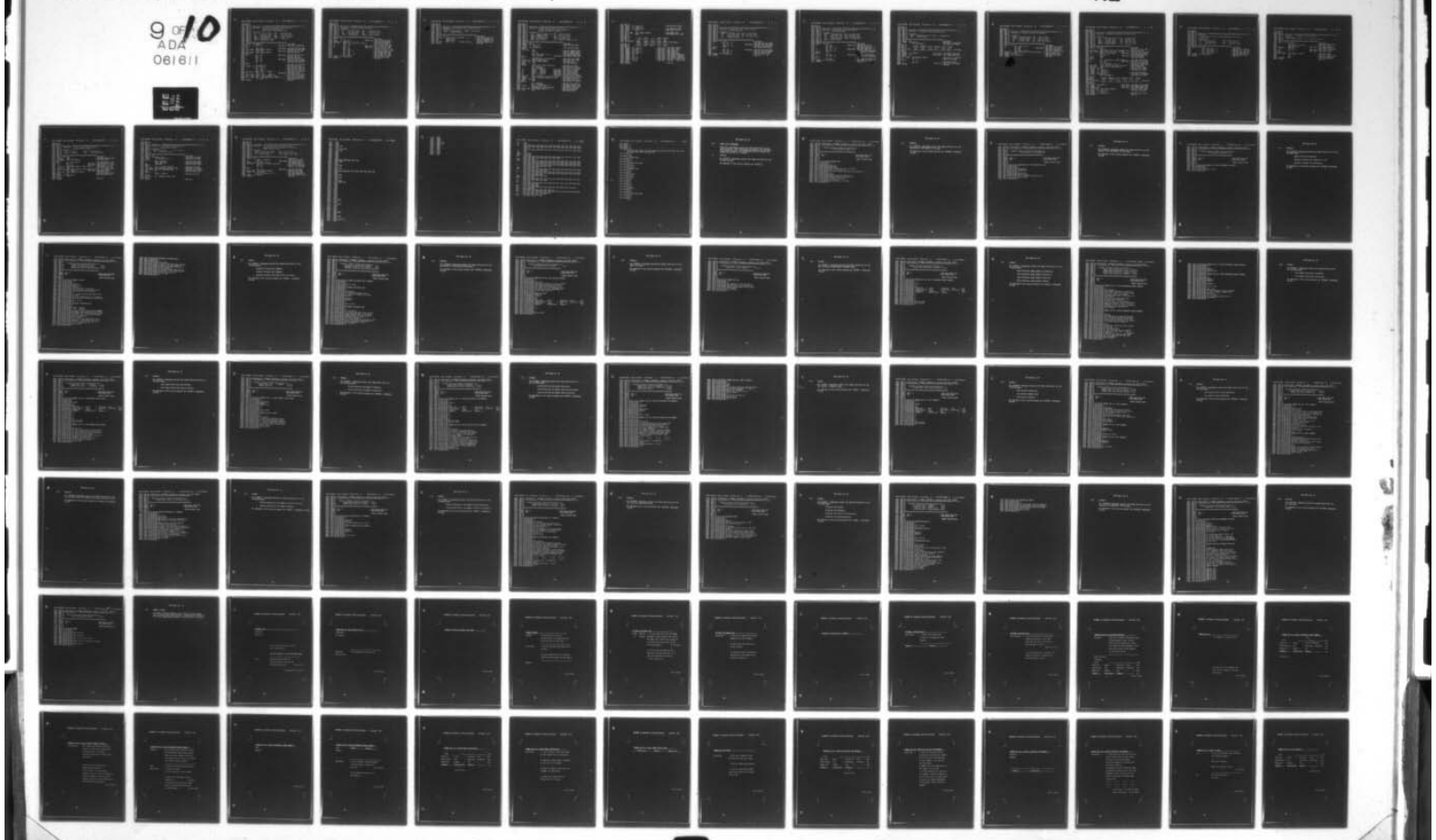
```
1704 REM **
1708 REM **
1712 REM **  Subroutine  ********************************************** 1 0 8
1716  REM **    Function >  To Input a character and verify
1720  REM **                for validity.
1724  REM **    Variables>
1728  REM **      R0$() > character bufr ! R4 > bufr indx <row>
1732  REM **      R8$() > micro cmnds    ! R5 > bufr indx <col>
1736  REM **      R4$() > translations   !
1740  REM **
1744 REM ************************************************
1748  DEFFN ' 108 (R0$(1))                                    : REM ENTRY
1752              STR(R8$(9),5,1) = R1$(R4,R5)                : REM STOR CURNT CHAR
1756  KEYIN       R0$(1), 1772, 1796                          : REM INPUT CHARACTER
1760  $GIO WAIT   /005(R8$(1),R8$(9))                         : REM INPUT WAIT
1764  GOTO        1756                                        : REM CONT WAIT
1768 REM **
1772  IF          R0$(1) < HEX(08)          THEN 1796 : REM INVALID INPUT
1776  $TRAN       (R0$() <1,1>, R4$()<257,16>) R              : REM CONV CTL CHARS
1780  IF          R0$(1)  > HEX(7F)         THEN 1796 : REM INVALID INPUT
1784  IF          R0$(1)  > HEX(1F)         THEN 1808 : REM VALID INPUT
1788  IF          R0$(1) <  HEX(06)         THEN 1808 : REM VALID INPUT
1792 REM **
1796  $GIO TONE   /005(R8$(7),R8$(9))                         : REM SOUND TONE
1800  GOTO        1756                                        : REM CONTINUE INPUT
1804 REM **
1808  RETURN                                                  : REM EXIT
```

```
1812 REM **
1816 REM **
1820  REM **  Subroutine  ********************************************** 1 0 9
1824  REM **    Function >  To Insert Character into buffer
1828  REM **                and process any control functions.
1832  REM **    Variables>
1836  REM **      RO$() > character buffer !
1840  REM **
1844  REM **********************************************
1848  DEFFN ' 109 (RO$(1))                                  : REM ENTRY
1852              RO = VAL(RO$(1))                          : REM CONV TO DEC
1856 REM **
1860 REM **         ! ERASE ! UNDRSCR ! EDIT ! ESCAPE ! CRLF ! OTHER !
1864 REM **         !-------!---------!------!--------!------!-------!
1868  ON RO GOTO     1884   ,  1888   , 1892 ,  1896  , 1900 , 1880  : REM BRNCH
1872 REM **         !-------!---------!------!--------!------!-------!
1876 REM **
1880  GOSUB ' 110 (RO$(1))                  : GOTO 1904 : REM ENTER CHARACTER
1884  GOSUB ' 111                           : GOTO 1904 : REM RUBOUT CHARACTER
1888  GOSUB ' 112                           : GOTO 1904 : REM UNDERSCORE CHAR.
1892  GOSUB ' 113                           : GOTO 1904 : REM EDIT BUFFER
1896  GOSUB ' 123                           : GOTO 1904 : REM INPUT COMPLETE
1900  GOSUB ' 124                           : GOTO 1904 : REM INSERT <CRLF>
1904  RETURN                                            : REM EXIT
```

AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U)
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

9 OF 10
ADA
061611



```
1908 REM **
1912 REM **
1916  REM **  Subroutine  ********************************************** 1 1 0
1920  REM **  Function >  To transfer character from character
1924  REM **              buffer to storage buffer.
1928  REM **              Store buffer on disk when full.
1932  REM **  Variables>
1936  REM **    R1     > crt pos <row>   ! R2     > crt pos <col>
1940  REM **    R4     > buf pos <row>   ! R5     > buf pos <col>
1944  REM **    R3     > bufr pointer    ! R6     > top lin of buf
1948  REM **    R1$()  > storage bufr    ! R0$()  > char. buffer
1952  REM **    R5$()  > terminator char! I1     > item id
1956  REM **    I2     > storg fnct code!
1960  REM **
1964  REM ***********************************************
1968  DEFFN ' 110 (R0$(1))                                    : REM ENTRY
1972  IF          R3 =  1                          THEN 2060 : REM FIRST CHARACTER
1976 REM **
1980  IF          R2 <> 59                         THEN 2000 : REM NOT TIME FOR TONE
1984  $GIO TONE   /005 (R8$(7),R8$(9))                      : REM END OF LINE TONE
1988  IF          R4 <> 16                         THEN 2000 : REM NOT END OF BUFR
1992  $GIO TONE   /005 (R8$(4),R8$(9))                      : REM PAUSE TONE
1996 REM **
2000  IF          R2 <> 64                         THEN 2040 : REM NOT END OF LINE
2004              R2 =  0                                   : REM SET CRT POS
2008              R5 =  0                                   : REM SET BUFFR POS
2012  IF          R1 <> 16                         THEN 2028 : REM NOT BOTM OF SCRN
2016              R1 =  0                                   : REM RESET FOR RAPARND
2020              R4 =  0                                   : REM SET BUFR POINTER
2024 REM **
2028              R1 =  R1 +  1                             : REM INCR CRT ROW INDX
2032  GOSUB ' 101 (R1,R2+1,0,0)                             : REM POSITION CUSOR
2036              R4 =  R4 + 1                              : REM INCR ROW INDX BUF
2040              R2 =  R2 + 1                              : REM INCR COL INDX CRT
2044              R5 =  R5 + 1                              : REM INCR COL INDX BUF
2048  $GIO LEFT   /005 (4009    ,R8$(9))                    : REM SPACE CUSOR LEFT
2052  IF          R2 <> 1                          THEN 2060 : REM SPACE IS OK
2056  $GIO RIGHT  /005  (4008    ,R8$(9))                   : REM SPACE CUSOR RIGHT
2060  MAT   COPY  R0$() <1,1> TO R1$() <R3,1>               : REM LOAD INTO BUFFER
2064  $GIO  CHAR  /005  (A0004008,R8$(9)) R0$() <1,1>       : REM DISPLAY CHARACTER
2068              R3  = R3 +  1                             : REM INCR BUFFER INDX
2072  RETURN                                                : REM EXIT
```

```
2076 REM **
2080 REM **
2084  REM **  Subroutine  ********************************************** 1 1 1
2088  REM **  Function > To Rubout current character.
2092  REM **  Variables >
2096  REM **    R1     > crt pos <row>  ! R2     > crt pos <col>
2100  REM **    R4     > buf pos <row>  ! R5     > buf pos <col>
2104  REM **    R3     > bufr pointer   ! R6     > top lin of buf
2108  REM **    R1$()  > storage bufr   ! RO$()  > char. buffer
2112  REM **
2116  REM ***********************************************
2120  DEFFN ' 111                                          : REM ENTRY
2124  IF          R3 =  1                       THEN 2176  : REM IF BUFR BEGINING
2128  GOSUB ' 101 (R1,R2,1,1)                              : REM ERASE CHARACTER
2132  MAT COPY    R5$() <1,1> TO R1$() <R3-1,1>            : REM CLR TEXT BUFFR
2136              R3 =  R3 -  1                            : REM DECRMNT BUFR INDX
2140  IF          R3 =  1                       THEN 2176  : REM IF BUFR BEGINING
2144  IF          R2 <> 1                       THEN 2164  : REM NOT BEG OF LINE
2148              R2 =  65                                 : REM SET TO END OF LINE
2152              R5 =  65                                 : REM SET TO END OF LINE
2156              R1 =  R1 -  1                            : REM DECRMNT ROW INDX
2160              R4 =  R4 -  1                            : REM DECRMNT ROW INDX
2164              R2 =  R2 -  1                            : REM DECRMNT COL INDX
2168              R5 =  R5 -  1                            : REM DECRMNT COL INDX
2172  GOSUB ' 101 (R1,R2,0,0)                             : REM POSITION CUSOR
2176  RETURN                                               : REM EXIT
```

```
2180 REM **
2184 REM **
2188  REM **  Subroutine  ********************************************** 1 1 2
2192  REM **  Function > To underscore the current character.
2196  REM **  Variables >
2200  REM **    R1$() > storage bufr    ! RO$() > char buffer
2204  REM **    R3    > buffer pointer !
2208  REM **
2212  REM ********************************************
2216  DEFFN ' 112                                          : REM ENTRY
2220  MAT COPY    R1$() <R3-1,1> TO RO$() <1,1>            : REM FETCH CHARACTER
2224  AND         (RO$(1),7F)                              : REM CLR UNDRSCR FLAG
2228  ADD         (RO$(1),80)                              : REM SET UNDRSCR FLAG
2232  MAT COPY    RO$() <1,1>    TO R1$() <R3-1,1>         : REM SAVE CHARACTER
2236  RETURN                                               : REM EXIT
```

```
2240 REM **
2244 REM **
2248  REM **  Subroutine  ********************************************** 1 1 3
2252  REM **  Function > To alow the storage buffer to be
2256  REM **             Edited. Editing can only be performed
2260  REM **             on data currently in buffer.
2264  REM **  Variables >
2268  REM **    IO   > value of numbr str ! R3  > buffer pointer
2272  REM **    R2$  > number string      ! R4  > buf indx <row>
2276  REM **    R1$()> storage buffer     ! R5  > buf indx <col>
2280  REM **    RO$()> character bufr     ! R4$()> translations
2284  REM **    R8$()> micro cmnds        ! I9  > length of numbr string
2288  REM **
2292  REM ********************************************
2296  DEFFN ' 113                                      : REM ENTRY
2300  GOSUB ' 101 (16,1,1,7)                           : REM POSITION CUSOR
2304  PRINT       "Edit ";
2308  GOSUB ' 101 (R1,R2,0,0)                          : REM POSITION CUSOR
2312 REM **
2316  INIT(20)    R2$                                  : REM CLR STORAGE BUFR
2320              I9 = 0                               : REM SET NUMBR LENGTH
2324              IO = 1                               : REM SET NUMBR VALUE
2328              R3 = (R4-1)*64 + R5 + 1              : REM CMPUTE CURNT INDX
2332  IF          R1$ = HEX(00)            THEN 2420   : REM REPEAT CUSR MOVE
2336 REM **
2340              STR(R8$(9),5,1) = R1$(R4,R5)         : REM STOR CURNT CHAR
2344  KEYIN       RO$(1), 2360, 2460                   : REM INPUT EDIT CMND
2348  $GIO CUSOR  /005 (R8$(1),R8$(9))                 : REM INPUT WAIT
2352  GOTO        2344                                 : REM CONTINUE WAIT
2356 REM **
2360  $TRAN       (RO$()<1,1>, R4$()<273,48>) R        : REM CONV CTL CHAR
2364              R1$ = HEX(20)                        : REM CLR REPT CUSR FLG
2368              RO = VAL(RO$(1))                     : REM CMPUT CHAR VALUE
2372  IF          RO$(1) <  HEX(0A)        THEN 2484   : REM VALID CTL CHAR
2376  IF          RO$(1)  > HEX(39)        THEN 2460   : REM INVALID INPUT
2380  IF          RO$(1)  > HEX(29)        THEN 2484   : REM VALID NUM CHAR
2384  IF          RO$(1)  > HEX(0A)        THEN 2412   : REM INVALID INPUT
2388  IF          R3     <> 2              THEN 2440   : REM NOT BEG OF BUFFER
2392  MAT COPY    Q2$()<1025,1024> TO R1$()<1,1024>    : REM RELOAD EDIT BUFFER
2396  $TRAN       (R1$()<1,1024>, R4$()<177,4>) R      : REM CONV 00 TO 20
2400  GOSUB ' 121 (R6)                                 : REM DISPLAY EDIT BUFR
2404  GOTO        2296                                 : REM BEGIN EDIT AGAIN
2408 REM **
2412              R1$ = HEX(00)                        : REM SET REPT CUSR FLG
2416              RO  = VAL(RO$(1)) - 10               : REM CMPUT BRANCH INDX
2420  KEYIN       RO$(1), 2360, 2420                   : REM REPEAT INTRUP CHK
2424              STR(R8$(9),5,1) = R1$(R4,R5)         : REM STOR CURNT CHAR
2428  $GIO CUSOR  /005 (R8$(1),R8$(9))                 : REM DISPLAY CUSOR
2432  GOTO        2484                                 : REM EXCUTE MOVE
```

```
2436 REM **
2440  GOSUB ' 101 (16,1,1,7)                                  : REM POSITION CUSOR
2444  PRINT       "Entry ";
2448  GOSUB ' 101 (R1,R2,0,0)                                 : REM POSITION CUSOR
2452  GOTO        2544                                        : REM BRANCH TO EXIT
2456 REM **
2460  $GIO TONE   /005 (R8$(7),R8$(9))                        : REM ERROR TONE
2464  GOTO        2316                                        : REM BRANCH EDIT INPUT
2468 REM **
2472 REM **
2476 REM **       !  LEFT ! RIGHT !   UP   !  DOWN ! CHANG !
2480 REM **       !-------!-------!-------!-------!-------!
2484  ON RO GOTO    2504   ,  2508 ,  2512 ,  2516 , 2520  ,
                    2524   ,  2528 ,  2532 ,  2536 , 2500  : REM BRANCH
2488 REM **       !-------!-------!-------!-------!-------!
2492 REM **       ! DELET ! INSRT ! BEGIN !  END  ! NUMBR !
2496 REM **
2500  GOSUB ' 120 (I0,I9)                  : GOTO 2340 : REM CMPUT NUMBER
2504  GOSUB ' 114 (I0*(-1))                : GOTO 2316 : REM CUSOR LEFT
2508  GOSUB ' 114 (I0)                     : GOTO 2316 : REM CUSOR RIGHT
2512  GOSUB ' 115 (I0*(-1))                : GOTO 2316 : REM CUSOR UP
2516  GOSUB ' 115 (I0)                     : GOTO 2316 : REM CUSOR DOWN
2520  GOSUB ' 116                          : GOTO 2316 : REM CHANGE CHARACTER
2524  GOSUB ' 117                          : GOTO 2316 : REM DELETE CHARACTER
2528  GOSUB ' 118 (I0)                     : GOTO 2316 : REM INSERT CHARACTER
2532  GOSUB ' 119 (1)                      : GOTO 2316 : REM SET TO BEG OF BFR
2536  GOSUB ' 119 (2)                      : GOTO 2316 : REM SET TO END OF BFR
2540 REM **
2544  RETURN                                             : REM EXIT
```

```
2548  REM **
2552  REM **
2556  REM **  Subroutine  ********************************************** 1 1 4
2560  REM **  Function > To position cusor left or right of
2564  REM **               current position.
2568  REM **  Variables >
2572  REM **    R0$() > character bufr ! R1 > crt pos <row>
2576  REM **    R5    > buf indx <col> ! R2 > crt pos <col>
2580  REM **    R3    > bufr indx linr ! R4 > buf indx <row>
2584  REM **    I0    > length of move !
2588  REM **
2592  REM ***********************************************
2596  DEFFN ' 114 (I0)                                      : REM ENTRY
2600              R2 = R2 + I0                              : REM CMPUT NEW COL INDX
2604  IF          R2 > 0                          THEN 2620 : REM NOT PAST LEFT MRGN
2608              R2 = R2 + 64                              : REM SCLE FOR RAPAROUND
2612  GOTO        2628                                      : REM BRANCH FOR SCROLL
2616 REM **
2620  IF          R2 < 65                         THEN 2628 : REM NOT PAST RGHT MRGN
2624              R2 = R2 - 64                              : REM SCLE FOR RAPAROUND
2628              R5 = R2                                   : REM SET NEW COL INDX
2632  GOSUB ' 101 (R1,R2,0,0)                               : REM POSITION CUSOR
2636  RETURN                                                : REM EXIT
```

```
2640 REM **
2644 REM **
2648  REM **  Subroutine  ********************************************** 1 1 5
2652  REM **  Function > To position cusor up or down of
2656  REM **               current position.
2660  REM **  Variables >
2664  REM **    RO$() > character bufr ! R1 > crt pos <row>
2668  REM **    R5    > buf indx <col> ! R2 > crt pos <col>
2672  REM **    R3    > bufr indx linr ! R4 > buf indx <row>
2676  REM **    IO    > length of move ! R1$()> storage bufr
2680  REM **    R8$() > micro cmnds    !
2684  REM **
2688  REM ***********************************************
2692  DEFFN ' 115 (IO)                                    : REM ENTRY
2696  IF          IO < 17                    THEN 2704    : REM WITHEN LIMITS
2700              IO = 15                                 : REM SET TO MAX VAL
2704              R4 = R4 + IO                            : REM CMPUT NEW ROW IDX
2708  IF          R4 > 0                     THEN 2724    : REM NOT OVER TOP
2712              R4 = R4 + 16                            : REM SCLE FOR RAPAROUND
2716  GOTO        2736                                    : REM BRANCH FOR SCROLL
2720 REM **
2724  IF          R4 < 17                    THEN 2736    : REM NOT OVER BOTTOM
2728              R4 = R4 - 16                            : REM SCLE FOR RAPAROUND
2732 REM **
2736              R1 = R4                                 : REM RESET CRT POSITION
2740  GOSUB ' 101 (R1,R2,0,0)                             : REM POSITION CUSOR
2744  RETURN                                              : REM EXIT
```

```
2748 REM **
2752 REM **
2756  REM **  Subroutine  ********************************************** 1 1 6
2760  REM **  Function > To Change current character only.
2764  REM **
2768  REM **  Variables >
2772  REM **    RO$() > character buffer ! R1$() > storage bufr
2776  REM **    R3    > buffer indx      !
2780  REM **
2784  REM **********************************************
2788  DEFFN ' 116                                        : REM ENTRY
2792  GOSUB ' 108 (RO$(1))                               : REM ENTER A CHARACTER
2796              RO = VAL(RO$(1))                       : REM CONV TO AN INDX
2800 REM **
2804 REM **        ! ERASE ! UNDRSCR ! EDIT ! ESCAPE ! CRLF ! OTHER !
2808 REM **        !-------!---------!------!--------!------!-------!
2812  ON RO GOTO    2836  ,  2828   , 2840 , 2840   , 2836 , 2824  : REM BRNCH
2816 REM **        !-------!---------!------!--------!------!-------!
2820 REM **
2824  GOSUB ' 110 (RO$(1))                 : GOTO 2848 : REM INSERT CHARACTER
2828  GOSUB ' 112                          : GOTO 2848 : REM UNDERSCORE CHAR
2832 REM **
2836  $GIO TONE   /005 (R8$(7),R8$(9))                  : REM INVALID CHARACTER
2840              IO = 0                                 : REM SET INDX END
2844 REM **
2848              IO = IO - 1                            : REM DECRMT INDEX
2852  IF          IO > 0                      THEN 2792  : REM CONT CHAR CHANGE
2856  RETURN                                             : REM EXIT
```

```
2860 REM **
2864 REM **
2868  REM **  Subroutine  ********************************************** 1 1 7
2872  REM **  Function > To Delete a string of characters.
2876  REM **
2880  REM **  Variables >
2884  REM **    R1$() > storage buffer ! R0  > spcs in bufr
2888  REM **    R9$() > scratch buffer ! I0  > spcs to delete
2892  REM **    R3    > buffer index   ! R1  > crt pos <row>
2896  REM **    R2    > crt pos <col>  ! R6  > top line of buf
2900  REM **
2904  REM ***********************************************
2908  DEFFN ' 117                                          : REM ENTRY
2912              R0 = 1024 - R3 + 1                       : REM CMPUT SPCE AVIABLE
2916  IF          I0 < R0                        THEN 2924 : REM SPACE AVAIABLE
2920              I0 = R0                                  : REM SET TO MAX SPACE
2924              I9 = 1024 - R3 + 1                       : REM CMPUT TRNSFR SPACE
2928  MAT COPY    R1$()<R3+I0,19-I0> TO R1$()<R3,I9>       : REM CLR BUFR TEXT
2932  GOSUB ' 121 (R6)                                     : REM RE-WRITE SCREEN
2936  GOSUB ' 101 (R1,R2,0,0)                              : REM POSITION CUSOR
2940  RETURN                                               : REM EXIT
```

```
2944 REM **
2948 REM **
2952  REM **  Subroutine  ********************************************** 1 1 8
2956  REM **  Function > To Insert characters into the buffer.
2960  REM **
2964  REM **  Variables >
2968  REM **    R0    > max spcs to insrt! R1  > crt pos <row>
2972  REM **    R2    > crt pos <col>    ! R3  > buffer indx
2976  REM **    R1$() > storage buffer   ! R4  > buf pos <row>
2980  REM **    R2$() > search result    ! R5  > buf pos <col>
2984  REM **    R9$() > scratch buffer   ! I0  > spcs to insrt
2988  REM **
2992  REM ********************************************
2996  DEFFN ' 118 (I0)                                    : REM ENTRY
3000              R0 = 16                                 : REM SET TO LAST LINE
3004  MAT COPY   -R1$()<((R0-1)*64+1,64> TO R9$()<1,64>: REM REVERSE COPY
3008  MAT SEARCH  R9$()<1,64>, <> R5$(1) TO R2$()       : REM SRCH FOR NON BLANK
3012  IF          R2$(1) <> HEX(0000)       THEN 3032 : REM NON BLANK FOUND
3016              R0 = R0 - 1                           : REM DECRMNT ROW INDX
3020  IF          R0 = 0                    THEN 3032 : REM BLANK BUFFER
3024  GOTO        3004                                  : REM CONT SEARCH
3028 REM **
3032              R0 = (16-R0)*64 + FNR(VAL(R2$(1)))    : REM CMPUT FREE SPACE
3036  IF          I0 < R0                   THEN 3044 : REM SPACE IS AVAIABLE
3040              I0 = R0 - 1                           : REM SET TO MAX SPACE
3044              I9 = (1024-R0) - R3 + 2               : REM CMPUT TRNSFR SPACE
3048  MAT COPY    -R1$()<R3,I9> TO -R1$()<R3+I0,I9>     : REM REVERSE COPY
3052  MAT COPY    R5$()<1,1> TO R1$()<R3,I0>            : REM CLR INSRT SPAC
3056  GOSUB ' 121 (R6)                                  : REM RE-WRITE SCREEN
3060  GOSUB ' 101 (16,1,1,7)                            : REM POSITION CUSOR
3064  PRINT       "Insert ";
3068  GOSUB ' 101 (R1,R2,0,0)                           : REM POSITION CUSOR
3072  GOSUB ' 108 (R0$(1))                              : REM ENTER A CHARACTER
3076              R0 = VAL(R0$(1))                      : REM CONV TO AN INDX
3080 REM **
3084 REM **       ! ERASE ! UNDRSCR ! EDIT ! ESCAPE ! CRLF ! OTHER !
3088 REM **       !-------!---------!------!--------!------!-------!
3092  ON R0 GOTO    3108  ,  3112   , 3120 ,  3120  , 3116 , 3104  : REM BRNCH
3096 REM **       !-------!---------!------!--------!------!-------!
3100 REM **
3104  GOSUB ' 110 (R0$(1))                : GOTO 3072 : REM SAVE CHAR IN BUFR
3108  GOSUB ' 111                         : GOTO 3072 : REM RUBOUT CHARACTER
3112  GOSUB ' 112                         : GOTO 3072 : REM UNDERSCORE CHAR.
3116  $GIO TONE   /005 (R8$(7),R8$(9))                  : REM EXIT INSERT MODE
3120  GOSUB ' 101 (16,1,1,7)                            : REM POSITION CUSOR
3124  PRINT       "Edit "
3128  GOSUB ' 101 (R1,R2,0,0)                           : REM POSITION CUSOR
3132  RETURN                                            : REM EXIT
```

```
3136 REM **
3140 REM **
3144 REM **  Subroutine  ********************************************** 1 1 9
3148 REM **  Abstract >
3152 REM **    To position the cusor at the begining or end of the
3156 REM **    text buffer.
3160 REM **  Variables >
3164 REM **    I9 > position index          ! R6 > top lin on crt
3168 REM **    R1   > crt pos <row>         ! R2   > crt pos <col>
3172 REM **    R4   > bufr pos <row>        ! R5   > bufr pos <col>
3176 REM **
3180 REM ***********************************************
3184  DEFFN ' 119 (I9)                                    : REM ENTRY
3188              R1 = (I9-1)*15 +  1                     : REM SET CRT  ROW POS
3192              R2 = (I9-1)*63 +  1                     : REM SET CRT  COL POS
3196              R4 = (I9-1)*15 +  1                     : REM SET BUFR ROW POS
3200              R5 = (I9-1)*63 +  1                     : REM SET BUFR COL POS
3204  GOSUB ' 101 (R1,R2,0,0)                            : REM POSITION CUSOR
3208  RETURN                                              : REM EXIT
```

```
3212 REM **
3216 REM **
3220  REM **  Subroutine  ********************************************** 1 2 0
3224  REM **  Function > To compute the cycle number.
3228  REM **  Variables >
3232  REM **    R2$ > number string  ! IO > length of string
3236  REM **
3240  REM ********************************************
3244  DEFFN ' 120 (IO,I9)                                    : REM ENTRY
3248              I9 = I9 + 1                                : REM INCR NUMBR INDX
3252              STR(R2$,I9,1) = RO$(1)                     : REM INSRT DIGIT
3256  IF          NUM(R2$) <> 64              THEN 3276      : REM INVALID NUMBER
3260  CONVERT     R2$ TO IO                                  : REM CMPUT NUMRC VALUE
3264  IF          IO > 64                     THEN 3276      : REM ILLEAGL VALUE
3268  IF          IO > 0                      THEN 3292      : REM LEAGL VALUE
3272 REM **
3276  $GIO TONE   /005 (R8$(7),R8$(9))                       : REM INPUT ERROR
3280              IO =  1                                    : REM SET TO DEFALT VAL
3284              I9 =  0                                    : REM RESET NUMBR LEN
3288  INIT(20)    R2$                                        : REM CLR NUMBR BUFR
3292  RETURN                                                 : REM EXIT
```

```
3296 REM **
3300 REM **
3304  REM **  Subroutine  ********************************************** 1 2 1
3308  REM **  Function >  To roll screen when position is
3312  REM **              advanced off of screen.
3316  REM **  Variables >
3320  REM **    R9     > scratch        ! R1$() > storage bufr
3324  REM **    R9$() > scratch         ! R6    > top of pag indx
3328  REM **
3332  REM **********************************************
3336  DEFFN ' 121 (R6)                                        : REM ENTRY
3340  INIT(2E)    R9$()                                       : REM LOAD SCRATCH BUFR
3344  $GIO HOME   /005 (4001,R8$(9))                          : REM HOME CUSOR
3348  REM **
3352  FOR         R9 = R6 TO R6+15                            : REM BEG REWRITE LOOP
3356   IF         R9 <= 0                        THEN 3364    : REM OUTSIDE PAG LIMIT
3360   IF         R9 <= 16                       THEN 3372    : REM INSIDE PAG LIMITS
3364   $GIO FILL  /005 (A000 ,R8$(9)) R9$()<1,64>              : REM PRINT FILL LINE
3368   GOTO       3376                                        : REM CONT OUTPUT LOOP
3372   $GIO PRINT /005 (A000 ,R8$(9)) R1$()<(R9-1)*64+1,64>    : REM PRINT TEXT
3376   IF         R9 = R6+15                     THEN 3384    : REM NULL ADVANCE
3380   $GIO CRLF  /005 (400D400A ,R8$(9))                      : REM PRINT CRLF
3384  NEXT        R9                                          : REM CONT REWRITE LOOP
3388  GOSUB ' 101 (16,1,1,7)                                  : REM POSITION CUSOR
3392  PRINT       "Edit  ";
3396  REM **
3400  RETURN                                                  : REM EXIT
```

```
3404 REM **
3408 REM **
3412  REM **  Subroutine  ********************************************** 1 2 3
3416  REM **  Function > To set End of Data Flag and
3420  REM **               Save buffer on disk.
3424  REM **  Variables >
3428  REM **    R2$  > eof flag         ! I1 > item id
3432  REM **    I2   > strge fnct code !
3436  REM **
3440  REM **********************************************
3444  DEFFN ' 123                                          : REM ENTRY
3448  GOSUB ' 101 (16,1,1,18)                              : REM POSITION CUSOR
3452  PRINT       "System Working... ";                    : REM PRINT SYS BUSY
3456  REM **
3460              R0$ = G1$(D2+1)                          : REM FETCH BEG ADDRS
3464              R3$ = HEX(0FFF)                          : REM DEFN ADRS MASK
3468  AND         (R0$,R3$)                                : REM CLR VOL ID BYTE
3472  REM **
3476  FOR         R0 = 1 TO 1024 STEP 256                  : REM BEG SAVE LOOP
3480   MAT COPY   R1$()<R0,256> TO G9$()<1,256>            : REM LOAD I/O BUFFER
3484   DATA SAVE  BA T #3, (R0$,R0$) G9$()                 : REM SAVE TEXT ON DSK
3488  NEXT        R0                                       : REM CONT SAVE LOOP
3492  REM **
3496              R0$(1) = HEX(FF)                         : REM SET EOF FLAG
3500  REM **
3504  LOAD DC T   #1, "ATH3LSTP" 1500, 9999
3508  REM **
3512  RETURN                                               : REM EXIT
```

```
3516 REM **
3520 REM **
3524  REM **  Subroutine  ********************************************** 1 2 4
3528  REM **  Function >  To position cusor and buffer pointer
3532  REM **              such that the next character entered will
3536  REM **              appear on next line.
3540  REM **  Variables >
3544  REM **    RO$() > character buffer     ! R2 > crt pos <col>
3548  REM **    R5 > buffer indx <col>       ! R4 > buffer indx <row>
3552  REM **    R1 > crt pos <row>           ! R3 > linear bufr indx
3556  REM **
3560  REM ************************************************
3564  DEFFN ' 124                                          : REM ENTRY
3568              RO$(1) = HEX(20)                         : REM CLR INPUT BUFR
3572  IF            R2 = 64                    THEN 3596   : REM END OF LINE
3576  GOSUB ' 101 (R1,R2+1,1,64-R2)                        : REM ERASE SCRN TEXT
3580  MAT COPY    RO$()<1,1> TO R1$()<R3,64-R2>            : REM ERASE BUFR TEXT
3584              R2 = 63                                  : REM SET TO LINE END
3588              R5 = 63                                  : REM SET TO LINE END
3592 REM **
3596  IF            R4 <> 16                   THEN 3604   : REM NOT LAST LINE
3600  $GIO TONE   /005 (R8$(4),R8$(9))                    : REM WARNING TONE
3604  GOSUB ' 101 (R1,R2,0,0)                              : REM POSITION CUSOR
3608              R3 = (R4-1)*64 + R5 + 1                  : REM CMPUT BUFR POS
3612  GOSUB ' 110 (RO$(1))                                 : REM SAVE CHAR IN BUFR
3616  RETURN                                               : REM EXIT
```

AUTHOR System: Phase 3 - ATH3EDTP - # MAP

```
1500  - 3504
1540  - 1548
1668  - 1668
1756  - 1764 1800
1772  - 1756
1796  - 1756 1772 1780
1808  - 1784 1788
1880  - 1868
1884  - 1868
1888  - 1868
1892  - 1868
1896  - 1868
1900  - 1868
1904  - 1880 1884 1888 1892 1896 1900
2000  - 1980 1988
2028  - 2012
2040  - 2000
2060  - 1972 2052
2164  - 2144
2176  - 2124 2140
2296  - 2404
2316  - 2464 2504 2508 2512 2516 2520 2524 2528 2532 2536
2340  - 2500
2344  - 2352
2360  - 2344 2420
2412  - 2384
2420  - 2332 2420
2440  - 2388
2460  - 2344 2376
2484  - 2372 2380 2432
2500  - 2484
2504  - 2484
2508  - 2484
2512  - 2484
2516  - 2484
2520  - 2484
2524  - 2484
2528  - 2484
2532  - 2484
2536  - 2484
2544  - 2452
2620  - 2604
2628  - 2612 2620
2704  - 2696
2724  - 2708
2736  - 2716 2724
2792  - 2852
2824  - 2812
2828  - 2812
2836  - 2812
2840  - 2812
2848  - 2824 2828
2924  - 2916
3004  - 3024
3032  - 3012 3020
3044  - 3036
3072  - 3104 3108 3112
3104  - 3092
```

```
3108   - 3092
3112   - 3092
3116   - 3092
3120   - 3092
3276   - 3256 3264
3292   - 3268
3364   - 3356
3372   - 3360
3376   - 3368
3384   - 3376
3596   - 3572
3604   - 3596
(9999) - 3504
```

AUTHOR System: Phase 3 - ATH3EDTP - V MAP

```
D2     - 3460
E1     - 1644
I0     - 2324 2500 2504 2508 2512 2516 2528 2596 2600 2692 2696 2700 2704 2840
         2848 2852 2916 2920 2928 2996 3036 3040 3048 3052 3244 3260 3264 3268
         3280
I9     - 2320 2500 2924 2928 3044 3048 3184 3188 3192 3196 3200 3244 3248 3252
         3284
Q1$(   - 3460
Q2$(   - 1672 2392
Q9$(   - 3480 3484
R0     - 1852 1868 2368 2416 2484 2796 2812 2912 2916 2920 3000 3004 3016 3020
         3032 3036 3040 3044 3076 3092 3476 3480 3488
R0$    - 3460 3468 3484
R0$(   - 1540 1544 1548 1664 1668 1748 1756 1772 1776 1780 1784 1788 1848 1852
         1880 1968 2060 2064 2220 2224 2228 2232 2344 2360 2368 2372 2376 2380
         2384 2416 2420 2792 2796 2824 3072 3076 3104 3252 3496 3568 3580 3612
R1     - 1620 1696 2012 2016 2028 2032 2128 2156 2172 2308 2448 2632 2736 2740
         2936 3068 3128 3188 3204 3576 3604
R1$    - 1648 2332 2364 2412
R1$(   - 1616 1656 1672 1676 1752 2060 2132 2220 2232 2340 2392 2396 2424 2928
         3004 3048 3052 3372 3480 3580
R2     - 1624 1696 1980 2000 2004 2032 2040 2052 2128 2144 2148 2164 2172 2308
         2448 2600 2604 2608 2620 2624 2628 2632 2740 2936 3068 3128 3192 3204
         3572 3576 3580 3584 3604
R2$    - 1652 2316 3252 3256 3260 3288
R2$(   - 3008 3012 3032
R3     - 1628 1972 2060 2068 2124 2132 2136 2140 2220 2232 2328 2388 2912 2924
         2928 3044 3048 3052 3580 3608
R3$    - 3464 3468
R4     - 1632 1752 1988 2020 2036 2160 2328 2340 2424 2704 2708 2712 2724 2728
         2736 3196 3596 3608
R4$(   - 1676 1776 2360 2396
R5     - 1636 1752 2008 2044 2152 2168 2328 2340 2424 2628 3200 3588 3608
R5$(   - 2132 3008 3052
R6     - 1640 1680 2400 2932 3056 3336 3352 3376
R8$(   - 1752 1760 1796 1984 1992 2048 2056 2064 2340 2348 2424 2428 2460 2836
         3116 3276 3344 3364 3372 3380 3600
R9     - 3352 3356 3360 3372 3376 3384
R9$(   - 1660 3004 3008 3340 3364
```

```
0000 DEFFN' 0
0001 DEFFN' 1
0002 DEFFN' 2
0003 DEFFN' 3
???? DEFFN' 101
       - 1688 1696 2032 2128 2172 2300 2308 2440 2448 2632 2740 2936 3060 3068
         3120 3128 3204 3388 3448 3576 3604
1612 DEFFN' 107
       - 1536
1748 DEFFN' 108
       - 1540 2792 3072
1848 DEFFN' 109
       - 1544
1968 DEFFN' 110
       - 1880 2824 3104 3612
2120 DEFFN' 111
       - 1884 3108
2216 DEFFN' 112
       - 1888 2828 3112
2296 DEFFN' 113
       - 1892
2596 DEFFN' 114
       - 2504 2508
2692 DEFFN' 115
       - 2512 2516
2788 DEFFN' 116
       - 2520
2908 DEFFN' 117
       - 2524
2996 DEFFN' 118
       - 2528
3184 DEFFN' 119
       - 2532 2536
3244 DEFFN' 120
       - 2500
3336 DEFFN' 121
       - 1680 2400 2932 3056
3444 DEFFN' 123
       - 1896
3564 DEFFN' 124
       - 1900
```

6.0 FORMAT DATA SUBPROGRAMS

These data subprograms provide the format definitions for the learning module. These subprograms are loaded by the 'AUTHOR/3' construction routines. For instructions on how to examine the formats see section 6.0 in Volume 1.

6.1 ATH1Ø1.Ø

The 'ATH1Ø1.Ø' subprogram contains the format definition for the Introduction page.

The remainder of this section presents the 'ATH1Ø1.Ø'.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH101.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 1 .0
9020  REM **
9030  REM **    ATH101.0 contains format descriptions for
9040  REM **              - The Introduction Page (1.0)
9050  REM **
9060  REM ***********************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,12,00,"INTRODUCTION"
9140  DATA 01,06,12,05," "
9150  DATA 01,18,38,04," "
9160  DATA 02,06,08,00,"Learning"
9170  DATA 03,06,09,00,"Objective"
9180  DATA 07,24,35,00,"This lesson will help you learn"
9190  DATA 08,24,35,00,"easily and quickly."
9200  DATA 10,24,45,02,"For best results, follow the directions."
9210  DATA 11,24,37,04," "
9220  DATA 12,06,15,00,"Index"
9230  DATA 12,24,40,00,"When you need an index to"
9240  DATA 13,24,40,00,"locate practice exercises for"
9250  DATA 14,24,40,00,"refresher training..."
9260  DATA 14,53,11,00,"Go to :indx"
9270  DATA 16,41,23,00,"To continue Go to :nxt0"
9280  DATA 00,00,00,09," "
```

6.2 ATH1Ø2.Ø

The 'ATH1Ø2.Ø' subprogram contains the format definition for the Importance of this Program to You page.

The remainder of this section presents the 'ATH1Ø2.Ø' subprogram listing.

```
9000  %## Data Subprogram > ATH102.0 (ATH102.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 2 .0
9020  REM **
9030  REM **    ATH102.0 contains format descriptions for
9040  REM **              - The Importance of Program (2.0)
9050  REM **
9060  REM ***********************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,33,00,"IMPORTANCE OF THIS PROGRAM TO YOU"
9140  DATA 01,06,33,05," "
9150  DATA 01,39,17,04," "
9160  DATA 02,06,10,00,"Importance"
9170  DATA 03,06,15,00,"of Symbols"
9180  DATA 08,24,37,04," "
9190  DATA 09,06,09,00,"How Skill"
9200  DATA 10,06,12,00,"Will be Used"
9210  DATA 09,24,37,00,"This program will help you to:"
9220  DATA 16,53,11,00,"Go to :nxt0"
9230  DATA 00,00,00,09," "
```

6.3 ATH1Ø3.Ø

The 'ATH1Ø3.Ø' subprogram contains the format definition for the Example of How the Symbols Are Used page.

The remainder of this section presents the 'ATH1Ø3.Ø' subprogram listing.

```
9000  %## Data Subprogram > ATH103.0 (ATH103.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 3 .0
9020  REM **
9030  REM **    ATH103.0 contains format descriptions for
9040  REM **              - The Example of Symbols (3.0)
9050  REM **
9060  REM ***********************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,36,00,"EXAMPLE OF HOW THE SYMBOLS ARE USED"
9140  DATA 01,06,36,05," "
9150  DATA 01,42,14,04," "
9160  DATA 16,53,11,00,"Go to :nxt0"
9170  DATA 00,00,00,09," "
```

6.4 ATH1Ø4.Ø

The 'ATH1Ø4.Ø' subprogram contains the format definitions for the following pages:

- Symbol Pre-Test (Directions)
- Optional Criterion Test (Choose #1 or #2)
- Optional Criterion Test (Directions)

The remainder of this section presents the 'ATH1Ø4.Ø' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH104.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ********************************************* 0 4 .0
9020  REM **
9030  REM **    ATH104.0 contains format descriptions for
9040  REM **         - Symbol Pre-Test; Directions          (4.1)
9050  REM **         - Symbol Pre-Test; End of Test         (4.2)
9060  REM **         - Optional Criterion Test; Directions (5.1)
9070  REM **
9080  REM ***********************************************
9090  REM **
9100  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9110  GOSUB ' 050                                        : REM RETURN JUMP
9120  REM **
9130  REM **                                               FORMAT DESCRIPTIONS
9140  REM **
9150  DATA 01,06,15,00,"SYMBOL PRETEST"
9160  DATA 01,06,15,05," "
9170  DATA 01,21,35,04," "
9180  DATA 02,06,07,00,"Purpose"
9190  DATA 06,06,10,00,"Directions"
9200  DATA 12,06,08,00,"Symbols"
9210  DATA 02,24,40,00,"The pretest will help you"
9220  DATA 03,24,40,00,"decide whether you should try the"
9230  DATA 04,24,40,00,"criterion test at this time."
9240  DATA 05,24,37,04," "
9250  DATA 06,24,40,00,"1. You may already know some or all"
9260  DATA 07,24,40,00,"of these"
9270  DATA 09,24,40,00,"2. Look through the list of symbols"
9280  DATA 10,24,40,00,"below and see how many you can define."
9290  DATA 11,24,37,04," "
9300  DATA 16,53,11,00,"Go to :nxt0"
9310  DATA 00,00,00,09," "
9320  DATA 01,06,23,00,"OPTIONAL CRITERION TEST"
9330  DATA 01,06,23,05," "
9340  DATA 01,29,27,04," "
9350  DATA 02,06,22,00,"Choose  Item #1"
9360  DATA 03,06,22,00,"  Or    Item #2"
9370  DATA 02,24,40,00,"1. If you think that there are THREE"
9380  DATA 03,24,40,00,"OR MORE of these symbols that YOU DO"
9390  DATA 04,24,40,00,"NOT KNOW, don't waste time taking the"
9400  DATA 05,24,40,00,"criterion test. Go directly to the "
9410  DATA 06,24,40,00,"learning program..."
9420  DATA 06,53,11,00,"Go to :skf2"
9430  DATA 08,24,40,00,"2. If you think you know all the"
9440  DATA 09,24,40,00,"symbols; or you missed only 1 or 2"
9450  DATA 10,24,40,00,"take the criterion test and see"
9460  DATA 11,24,40,00,"if you can pass it..."
9470  DATA 16,53,11,00,"Go to :nxt0"
9480  DATA 00,00,00,09," "
```

```
9490  DATA 01,06,23,00,"OPTIONAL CRITERION TEST"
9500  DATA 01,06,23,05," "
9510  DATA 01,29,27,04," "
9520  DATA 02,06,10,00,"Directions"
9530  DATA 02,24,35,00,"1. Recall the definition of each"
9540  DATA 03,24,40,00,"symbol on the next frame(s)."
9550  DATA 05,24,35,00,"2. Write each definition on a "
9560  DATA 06,24,15,00,"piece of paper."
9570  DATA 08,24,40,00,"3. You must recall the official "
9580  DATA 09,24,40,00,"definition of the symbol to be "
9590  DATA 10,24,40,00,"correct (as found in "
9600  DATA 16,53,11,00,"Go to :nxt0"
9610  DATA 00,00,00,09," "
```

6.5 ATH1Ø4.1

The 'ATH1Ø4.1' subprogram contains the format definitions for the following pages:

- Optional Criterion Test (Symbols)
- Optional Criterion Test (Answers)
- Optional Criterion Test (End-of-Test Directions)

The remainder of this section presents the 'ATH1Ø4.1' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH104.1)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 4 .1
9020  REM **
9030  REM **    ATH104.1 contains format descriptions for
9040  REM **        - Optional Criterion Test; Symbols    (5.2)
9050  REM **        - Optional Criterion Test; Answers    (5.3)
9060  REM **        - Optional Criterion Test; End of Test(5.4)
9070  REM **
9080  REM ******************************************************
9090  REM **
9100  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9110  GOSUB ' 050                                      : REM RETURN JUMP
9120  REM **
9130  REM **                                             FORMAT DESCRIPTIONS
9140  REM **
9150  DATA 01,06,33,00,"OPTIONAL CRITERION TEST: SYMBOLS"
9160  DATA 01,06,33,05," "
9170  DATA 01,39,17,04," "
9180  DATA 16,53,11,00,"Go to :nxt0"
9190  DATA 00,00,00,09," "
9200  DATA 01,06,23,00,"OPTIONAL CRITERION TEST"
9210  DATA 01,06,23,05," "
9220  DATA 01,29,27,04," "
9230  DATA 02,06,10,00,"Directions"
9240  DATA 02,24,26,00,"1. Check your answers now."
9250  DATA 03,24,31,00,"2. Put an X through your wrong"
9260  DATA 04,24,08,00,"answers."
9270  DATA 05,24,37,04," "
9280  DATA 06,06,08,00,"Symbols"
9290  DATA 06,36,07.00,"Answers"
9300  DATA 06,06,19,05," "
9310  DATA 06,26,30,05," "
9320  DATA 16,53,11,00,"Go to :nxt0"
9330  DATA 00,00,00,09," "
9340  DATA 01,06,23,00,"OPTIONAL CRITERION TEST"
9350  DATA 01,06,23,05," "
9360  DATA 01,29,27,04," "
9370  DATA 02,06,13,00,"Did you pass?"
9380  DATA 02,24,40,00,"1. Did you miss ## or less of the"
9390  DATA 03,24,40,00,"symbol definitions? If so, then"
9400  DATA 04,24,33,00,"learn the ones that you missed by"
9410  DATA 05,24,28,00,"studying them on :skb1 ..."
9420  DATA 06,48,16,00,"Then Go to :clos"
9430  DATA 08,24,40,00,"2. If you missed more than ## of the"
9440  DATA 09,24,40,00,"symbol definitions, complete this"
9450  DATA 10,24,28,00,"learning program - Now ..."
9460  DATA 16,53,11,00,"Go to :nxt0"
9470  DATA 00,00,00,09," "
```

6.6 ATH1Ø5.Ø

The 'ATH1Ø5.Ø' subprogram contains the format definition for the Organization of the Learning Program page.

The remainder of this section presents the 'ATH1Ø5.Ø' subprogram listing.

AUTHOR System: Phase 1 - ATH105.0 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH105.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 5 .0
9020  REM **
9030  REM **    ATH105.0 contains format descriptions for
9040  REM **        - The Organization of Program        (6.0)
9050  REM **
9060  REM ********************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,36,00,"ORGANIZATION OF THE LEARNING PROGRAM"
9140  DATA 01,06,36,05," "
9150  DATA 01,42,14,04," "
9160  DATA 02,06,08,00,"Overview"
9170  DATA 02,24,40,00,"The list of symbols has been divided"
9180  DATA 03,24,40,00,"into ## practice sets. You will"
9190  DATA 04,24,40,00,"learn each set separately, using"
9200  DATA 05,24,40,00,"the 'Steps for Easy Learning.' Then"
9210  DATA 06,24,40,00,"you will practice all ## symbols"
9220  DATA 07,24,40,00,"in the same exercise."
9230  DATA 08,24,37,04," "
9240  DATA 09,06,14,00,"Steps for Easy"
9250  DATA 10,06,08,00,"Learning"
9260  DATA 11,06,05,00,":dia0"
9270  DATA 11,25,35,04," "
9280  DATA 11,60,02,00,"No"
9290  DATA 12,03,60,00,"!Note the   !  !Learn       !  !Practice!  !Test     !   Did"
9300  DATA 13,03,60,00,"!Differences!  !the         !  !Defining!  !Yourself!   You"
9310  DATA 14,03,61,00,"!Among the  !  !symbol      !  !the     !  !         !   Pass"
9320  DATA 15,03,59,00,"!Symbols    !  !Definitions!  !Symbols !  !         !    ?"
9330  DATA 15,04,11,05," "
9340  DATA 15,19,11,05," "
9350  DATA 15,34,08,05," "
9360  DATA 15,46,08,04," "
9370  DATA 15,58,03,05," "
9380  DATA 16,53,11,00,"Go to :nxt0"
9390  DATA 00,00,00,09," "
```

6.7 ATH1Ø6.Ø

The 'ATH1Ø6.Ø' subprogram contains the format definition for the symbol list page.

The remainder of this section presents the 'ATH1Ø6.Ø' subprogram listing.

AUTHOR System: Phase 1 - ATH106.0 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .0
9020  REM **
9030  REM **    ATH106.0 contains format descriptions for
9040  REM **        - Symobl Set; List of Symbols       (7.1.0)
9050  REM **
9060  REM ***********************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,15,00,"SYMBOL SET #zz"
9140  DATA 01,06,15,05," "
9150  DATA 01,21,34,04," "
9160  DATA 02,24,40,00,"The symbols in this set are:"
9170  DATA 13,24,40,00,"You will learn to recognize and"
9180  DATA 14,24,40,00,"define these symbols in the next"
9190  DATA 15,24,40,00,"few frames."
9200  DATA 16,53,11,00,"Go to :nxt0"
9210  DATA 00,00,00,09," "
```

6.8 ATH1Ø6.1

The 'ATH1Ø6.1' subprogram contains the format definition for the Note Differences Among Symbols Flowchart page.

The remainder of this section presents the 'ATH1Ø6.1' subprogram listing.

AUTHOR System: Phase 1 - ATH106.1 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.1)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .1
9020  REM **
9030  REM **    ATH106.1 contains format descriptions for
9040  REM **        - Symobl Set; Diferences ; Diagram  (7.2.1)
9050  REM **
9060  REM ********************************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,49,00,"SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS"
9140  DATA 01,06,49,05," "
9150  DATA 01,55,01,04," "
9160  DATA 03,06,05,00,":dial"
9170  DATA 03,25,35,04," "
9180  DATA 03,60,02,00,"No"
9190  DATA 04,02,62,00,"!!Note the   !!  !Learn       !  !Practice!  !Test     !   Did"
9200  DATA 05,02,62,00,"!!Differences!!  !the         !  !Defining!  !Yourself!   You"
9210  DATA 06,02,62,00,"!!Among the  !!  !symbol      !  !the      !  !         !   Pass"
9220  DATA 07,02,62,00,"!!Symbols    !!  !Definitions!  !Symbols !  !         !    ?"
9230  DATA 07,03,12,05," "
9240  DATA 07,20,11,05," "
9250  DATA 07,35,08,05," "
9260  DATA 07,47,08,04," "
9270  DATA 07,59,03,05," "
9280  DATA 08,08,14,00,"*"
9290  DATA 09,03,40,00,"You are here"
9300  DATA 16,53,11,00,"Go to :nxt0"
9310  DATA 00,00,00,09," "
```

6.9 ATH1Ø6.2

The 'ATH1Ø6.2' subprogram contains the format definitions for the following pages:

- Note Differences Among Symbols (Introduction)
- Note Differences Among Symbols (Directions)
- Note Differences Among Symbols (Exercises)
- Note Differences Among Symbols (Answers)

The remainder of this section presents the 'ATH106.2' subprogram listing.

AUTHOR System: Phase 1 - ATH106.2 Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.2)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .2
9020  REM **
9030  REM **    ATH106.2 contains format descriptions for
9040  REM **        - Symbol Set; Diferences ; Intro    (7.2.2)
9050  REM **        - Symbol Set; Diferences ;Directions(7.2.3)
9060  REM **        - Symbol Set; Diferences ; Symbols  (7.2.4)
9070  REM **        - Symbol Set; Diferences ; Answers  (7.2.5)
9080  REM **
9090  REM ******************************************************
9100  REM **
9110  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9120  GOSUB ' 050                                        : REM RETURN JUMP
9130  REM **
9140  REM **                                               FORMAT DESCRIPTIONS
9150  REM **
9160  DATA 01,06,49,00,"SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS"
9170  DATA 01,06,49,05," "
9180  DATA 01,55,01,04," "
9190  DATA 02,06,12,00,"Introduction"
9200  DATA 02,24,40,00,"Each symbol represents a different"
9210  DATA 03,24,40,00,"thing, yet some of the symbols look"
9220  DATA 04,24,40,00,"very much alike. The symbols"
9230  DATA 05,24,40,00,"presented below may be confused with"
9240  DATA 06,24,40,00,"each other:"
9250  DATA 09,24,40,00,"Carefully note what makes the"
9260  DATA 10,24,40,00,"symbols above different."
9270  DATA 11,24,40,00,"At this point in the learning"
9280  DATA 12,24,40,00,"program, look only at the differences"
9290  DATA 13,24,40,00,"among the symbols. Later on in the"
9300  DATA 14,24,40,00,"program you will be given definitions"
9310  DATA 15,24,40,00,"associated with each symbol."
9320  DATA 16,53,11,00,"Go to :nxt0"
9330  DATA 00,00,00,09," "
9340  DATA 01,06,49,00,"SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS"
9350  DATA 01,06,49,05," "
9360  DATA 01,55,01,04," "
9370  DATA 02,06,07,00,"Purpose"
9380  DATA 02,24,40,00,"This exercise will help you to note"
9390  DATA 03,24,40,00,"the differences among those symbols"
9400  DATA 04,24,40,00,"that are often confused. After you"
9410  DATA 05,24,40,00,"finish with this exercise, you should"
9420  DATA 06,24,40,00,"not confuse similar symbols."
9430  DATA 07,24,37,04," "
9440  DATA 08,06,04,00,"Quiz"
9450  DATA 08,24,40,00,"1. Read all directions before going"
9460  DATA 09,06,10,00,"Directions"
9470  DATA 09,24,18,00,"to the next frame."
9480  DATA 10,24,40,00,"2. Look carefully at this symbol:"
9490  DATA 12,24,40,00,"3. Write the item number of each"
9500  DATA 13,24,40,00,"example of                    on paper."
9510  DATA 14,24,40,00,"4. Do not look back at this frame"
9520  DATA 15,24,40,00,"until you have finished the quiz"
9530  DATA 16,24,40,00,"for this symbol."
9540  DATA 16,53,11,00,"Go to :nxt2"
9550  DATA 00,00,00,09," "
```

```
9560  DATA 01,06,49,00,"SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS"
9570  DATA 01,06,49,05," "
9580  DATA 01,55,01,04," "
9590  DATA 02,06,20,00,"Quiz"
9600  DATA 03,06,20,00,"Symbols"
9610  DATA 16,53,11,00,"Go to :nxt0"
9620  DATA 00,00,00,09," "
9630  DATA 01,06,49,00,"SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS"
9640  DATA 01,06,49,05," "
9650  DATA 01,55,01,04," "
9660  DATA 02,06,20,00,"Quiz"
9670  DATA 03,06,20,00,"Answers"
9680  DATA 02,24,10,00,"Symbol"
9690  DATA 02,24,15,05," "
9700  DATA 02,42,10,00,"Examples"
9710  DATA 02,40,16,05," "
9720  DATA 16,53,11,00,"Go to :nxt0"
9730  DATA 06,24,37,04," "
9740  DATA 07,06,20,00,"What Now?"
9750  DATA 07,24,40,00,"If you frequently missed examples"
9760  DATA 08,24,40,00,"of these symbols, repeat this"
9770  DATA 09,24,40,00,"exercise . . ."
9780  DATA 09,53,11,00,"Go to :skb1"
9790  DATA 11,24,40,00,"If you identified most all of"
9800  DATA 12,24,40,00,"the symbols . . ."
9810  DATA 00,00,00,09," "
```

6.10 ATH1Ø6.3

The 'ATH1Ø6.3' subprogram contains the format definitions for the following pages:

- Learn Symbol Definitions (Flowchart)
- Learn Symbol Definitions (Directions)

The remainder of this section presents the 'ATH1Ø6.3' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.3)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .3
9020  REM **
9030  REM **    ATH106.3 contains format descriptions for
9040  REM **        - Symbol Set; Learn    ; Diagram    (7.3.1)
9050  REM **        - Symbol Set; Learn    ; Directions (7.3.2)
9060  REM **
9070  REM ******************************************************
9080  REM **
9090  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9100  GOSUB ' 050                                      : REM RETURN JUMP
9110  REM **
9120  REM **                                             FORMAT DESCRIPTIONS
9130  REM **
9140  DATA 01,06,42,00,"SYMBOL SET #zz: LEARN SYMBOL DEFINITIONS"
9150  DATA 01,06,42,05," "
9160  DATA 01,48,08,04," "
9170  DATA 03,06,05,00,":dia2"
9180  DATA 03,25,36,04," "
9190  DATA 03,61,02,00,"No"
9200  DATA 04,02,62,00,"!Note the    !  !!Learn      !!  !Practice!  !Test     !   Did"
9210  DATA 05,02,62,00,"!Differences!  !!the        !!  !Defining!  !Yourself!   You"
9220  DATA 06,02,62,00,"!Among the   !  !!symbol     !!  !the      !  !         !   Pass"
9230  DATA 07,02,62,00,"!Symbols     !  !!Definitions!!  !Symbols !  !         !    ?"
9240  DATA 07,03,11,05," "
9250  DATA 07,18,13,05," "
9260  DATA 07,35,08,05," "
9270  DATA 07,47,08,04," "
9280  DATA 07,59,03,05," "
9290  DATA 08,24,14,00,"*"
9300  DATA 09,19,15,00,"You are here"
9310  DATA 16,53,11,00,"Go to :nxt0"
9320  DATA 00,00,00,09," "
9330  DATA 01,06,42,00,"SYMBOL SET #zz: LEARN SYMBOL DEFINITIONS"
9340  DATA 01,06,42,05," "
9350  DATA 01,48,08,04," "
9360  DATA 02,06,15,00,"Directions"
9370  DATA 02,24,40,00,"1. Look carefully on the next frame"
9380  DATA 03,24,40,00,"at each symbol and its definition."
9390  DATA 05,24,40,00,"2. Repeat the definition to yourself"
9400  DATA 06,24,40,00,"as you look at the symbol."
9410  DATA 08,24,40,00,"3. Study the memory aid to help you"
9420  DATA 09,24,40,00,"remember the definition."
9430  DATA 11,24,40,00,"4. Study each symbol and its"
9440  DATA 12,24,40,00,"definition 2 or 3 times."
9450  DATA 16,53,11,00,"Go to :nxt0"
9460  DATA 00,00,00,09," "
```

6.11 ATH1Ø6.4

The 'ATH1Ø6.4' subprogram contains the format definitions for the following pages:

- Learn Symbol Definitions (Definitions)
- Learn Symbol Definitions (Rest Directions)

The remainder of this section presents the 'ATH1Ø6.4' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.4)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .4
9020  REM **
9030  REM **    ATH106.4 contains format descriptions for
9040  REM **         - Symbol Set; Learn    ; Symbols    (7.3.3)
9050  REM **         - Symbol Set; Rest                  (7.4.0)
9060  REM **
9070  REM ******************************************************
9080  REM **
9090  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9100  GOSUB ' 050                                        : REM RETURN JUMP
9110  REM **
9120  REM **                                               FORMAT DESCRIPTIONS
9130  REM **
9140  DATA 01,06,42,00,"SYMBOL SET #zz: LEARN SYMBOL DEFINITIONS"
9150  DATA 01,06,42,05," "
9160  DATA 01,48,08,04," "
9170  DATA 02,11,11,00,"Definition"
9180  DATA 02,31,08,00,"Symbol"
9190  DATA 02,48,11,00,"Memory Aid"
9200  DATA 02,04,22,05," "
9210  DATA 02,27,15,05," "
9220  DATA 02,43,20,05," "
9230  DATA 16,53,11,00,"Go to :nxt0"
9240  DATA 00,00,00,09," "
9250  DATA 01,06,21,00,"SYMBOL SET #zz: REST"
9260  DATA 01,06,21,05," "
9270  DATA 01,27,29,04," "
9280  DATA 03,06,15,00,"Directions"
9290  DATA 03,24,40,00,"-- Relax for a moment or two."
9300  DATA 04,24,40,00,"Get your mind off your study."
9310  DATA 06,24,40,00,"-- Think of something relaxing."
9320  DATA 08,24,40,00,"-- You will learn more quickly"
9330  DATA 09,24,40,00,"when you go back to your study"
9340  DATA 10,24,40,00,"if you rest."
9350  DATA 16,53,11,00,"Go to :nxt0"
9360  DATA 00,00,00,09," "
```

6.12 ATH1Ø6.5

The 'ATH1Ø6.5' subprogram contains the format definitions for the following pages:

- Practice Defining the Symbols (Flowchart)
- Practice Defining the Symbols (Directions)

The remainder of this section presents the 'ATH1Ø6.5' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.5)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .5
9020  REM **
9030  REM **    ATH106.5 contains format descriptions for
9040  REM **        - Symbol Set; Practice ; Diagram    (7.5.1)
9050  REM **        - Symbol Set; Practice ; Directions (7.5.2)
9060  REM **
9070  REM ****************************************************
9080  REM **
9090  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9100  GOSUB ' 050                                      : REM RETURN JUMP
9110  REM **
9120  REM **                                             FORMAT DESCRIPTIONS
9130  REM **
9140  DATA 01,06,47,00,"SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS"
9150  DATA 01,06,47,05," "
9160  DATA 01,53,03,04," "
9170  DATA 03,06,05,00,":dia3"
9180  DATA 03,25,36,04," "
9190  DATA 03,61,02,00,"No"
9200  DATA 04,02,62,00,"!Note the    !  !Learn       !  !!Practice!!  !Test     !   Did"
9210  DATA 05,02,62,00,"!Differences!  !the          !  !!Defining!!  !Yourself!   You"
9220  DATA 06,02,62,00,"!Among the   !  !symbol       !  !!the      !!  !         !   Pass"
9230  DATA 07,02,62,00,"!Symbols     !  !Definitions!  !!Symbols !!  !         !    ?"
9240  DATA 07,03,11,05," "
9250  DATA 07,18,11,05," "
9260  DATA 07,33,10,05," "
9270  DATA 07,47,08,04," "
9280  DATA 07,59,03,05," "
9290  DATA 08,37,14,00,"*"
9300  DATA 09,32,15,00,"You are here"
9310  DATA 16,53,11,00,"Go to :nxt0"
9320  DATA 00,00,00,09," "
9330  DATA 01,06,47,00,"SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS"
9340  DATA 01,06,47,05," "
9350  DATA 01,53,03,04," "
9360  DATA 02,06,20,00,"Directions"
9370  DATA 02,24,40,00,"1. Read all directions before"
9380  DATA 03,24,40,00,"you practice defining the symbols."
9390  DATA 04,24,40,00,"2. Try to recall the definition"
9400  DATA 05,24,40,00,"of each symbol in the exercise"
9410  DATA 06,24,40,00,"on frame :nxt0."
9420  DATA 07,24,40,00,"3. WRITE your first impression of"
9430  DATA 08,24,40,00,"the symbol's definition."
9440  DATA 09,24,40,00,"4. If the definition is difficult"
9450  DATA 10,24,40,00,"to remember, recall the memory aid"
9460  DATA 11,24,40,00,"first, then recall the definition."
9470  DATA 12,24,40,00,"5. Check your answer immediately"
9480  DATA 13,24,40,00,"in the answer section below the"
9490  DATA 14,24,40,00,"symbols."
9500  DATA 16,53,11,00,"Go to :nxt0"
9510  DATA 00,00,00,09," "
```

6.13 ATH1Ø6.6

The 'ATH1Ø6.6' subprogram contains the format definitions for the following pages:

- Practice Defining the Symbols (Exercises)
- Practice Defining the Symbols (Practice Directions)
- Practice Defining the Symbols (Rest Directions)

The remainder of this section presents the 'ATH1Ø6.6' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.6)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .6
9020  REM **
9030  REM **    ATH106.6 contains format descriptions for
9040  REM **        - Symbol Set; Practice ; Symbols    (7.5.3)
9050  REM **        - Symbol Set; Practice ; Directions (7.5.4)
9060  REM **        - Symbol Set; Rest                  (7.6.0)
9070  REM **
9080  REM ******************************************************
9090  REM **
9100  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9110  GOSUB ' 050                                      : REM RETURN JUMP
9120  REM **
9130  REM **                                             FORMAT DESCRIPTIONS
9140  REM **
9150  DATA 01,06,47,00,"SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS"
9160  DATA 01,06,47,05," "
9170  DATA 01,53,03,04," "
9180  DATA 02,06,20,00,"Practice"
9190  DATA 03,06,20,00,"Symbols"
9200  DATA 07,06,54,04," "
9210  DATA 08,32,11,00,"Definition"
9220  DATA 08,10,08,00,"Symbol"
9230  DATA 08,22,38,05," "
9240  DATA 08,06,15,05," "
9250  DATA 16,53,11,00,"Go to :nxt0"
9260  DATA 00,00,00,09," "
9270  DATA 01,06,47,00,"SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS"
9280  DATA 01,06,47,05," "
9290  DATA 01,53,03,04," "
9300  DATA 02,06,20,00,"Directions"
9310  DATA 02,24,40,00,"1. Keep practicing until you feel you"
9320  DATA 03,24,40,00,"know the definition of each symbol."
9330  DATA 04,24,40,00,"2. Practice defining the most"
9340  DATA 05,24,40,00,"difficult symbols for you more than"
9350  DATA 06,24,40,00,"the easier symbols."
9360  DATA 07,24,40,00,"3. Change the order of going through"
9370  DATA 08,24,40,00,"the practice symbol list as you study."
9380  DATA 09,24,40,00,"Examples of how you might vary the"
9390  DATA 10,24,40,00,"order of studying the symbol codes "
9400  DATA 11,24,40,00,"in the list:"
9410  DATA 12,24,40,00,"----->    <-----    ----->    <-----"
9420  DATA 13,24,40,00,"<-----    ----->    ----->    <-----"
9430  DATA 14,24,40,00,"----->    <-----    ----->    <-----"
9440  DATA 15,32,32,00,"To practice...  Go back to "
9450  DATA 15,59,05,00,":skb1"
9460  DATA 16,32,32,00,"After practicing...  Go to "
9470  DATA 16,59,05,00,":nxt0"
9480  DATA 00,00,00,09," "
```

```
9490  DATA 01,06,30,00,"SYMBOL SET #zz: WANT TO REST?"
9500  DATA 01,06,30,05," "
9510  DATA 01,36,20,04," "
9520  DATA 02,06,10,00,"Yes"
9530  DATA 10,06,10,00,"No"
9540  DATA 02,24,40,00,"If you are tired now, get your mind"
9550  DATA 03,24,40,00,"off studying symbols."
9560  DATA 05,24,40,00,"Relax for a minute!"
9570  DATA 07,24,40,00,"After your minute's rest..."
9580  DATA 08,53,11,00,"Go to :nxt0"
9590  DATA 09,24,37,04," "
9600  DATA 10,24,40,00,"If you do not want to"
9610  DATA 11,24,40,00,"rest now..."
9620  DATA 16,53,11,00,"Go to :nxt0"
9630  DATA 00,00,00,09," "
```

6.14 ATH1Ø6.7

The 'ATH1Ø6.7' subprogram contains the format definition for the Test Yourself (Flowchart) page.

The remainder of this section presents the 'ATH1Ø6.7' subprogram listing.

AUTHOR System: Phase 1 - ATH106.7 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.7)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .7
9020  REM **
9030  REM **    ATH106.7 contains format descriptions for
9040  REM **        - Symbol Set; Test Yourself Diagram (7.7.1)
9050  REM **
9060  REM ******************************************************
9070  REM **
9080  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                      : REM RETURN JUMP
9100  REM **
9110  REM **                                             FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,30,00,"SYMBOL SET #zz: TEST YOURSELF"
9140  DATA 01,06,30,05," "
9150  DATA 01,36,20,04," "
9160  DATA 03,06,05,00,":dia4"
9170  DATA 03,25,36,04," "
9180  DATA 03,61,02,00,"No"
9190  DATA 04,02,62,00,"!Note the    !  !Learn       !  !Practice!  !!Test     !!   Did"
9200  DATA 05,02,62,00,"!Differences!  !the          !  !Defining!  !!Yourself!!   You"
9210  DATA 06,02,62,00,"!Among the   !  !symbol       !  !the     !  !!          !!   Pass"
9220  DATA 07,02,62,00,"!Symbols     !  !Definitions!  !Symbols !  !!          !!    ?"
9230  DATA 07,03,11,05," "
9240  DATA 07,18,11,05," "
9250  DATA 07,33,08,05," "
9260  DATA 07,45,10,05," "
9270  DATA 07,59,03,05," "
9280  DATA 08,50,14,00,"*"
9290  DATA 09,45,15,00,"You are here"
9300  DATA 16,53,11,00,"Go to :nxt0"
9310  DATA 00,00,00,09," "
```

6.15 ATH1Ø6.8

The 'ATH1Ø6.8' subprogram contains the format definitions for the following pages:

- Test Yourself (Directions)
- Test Yourself (Memory Aids)
- Test Yourself (Answers)

The remainder of this section presents the 'ATH1Ø6.8' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.8)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .8
9020  REM **
9030  REM **    ATH106.8 contains format descriptions for
9040  REM **        - Symbol Set; Test Yourself Direct  (7.7.2)
9050  REM **        - Symbol Set; Test Yourself Mem Aid (7.7.3)
9060  REM **        - Symbol Set; Test Yourself Answers (7.7.4)
9070  REM **
9080  REM ******************************************************
9090  REM **
9100  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9110  GOSUB ' 050                                      : REM RETURN JUMP
9120  REM **
9130  REM **                                             FORMAT DESCRIPTIONS
9140  REM **
9150  DATA 01,06,30,00,"SYMBOL SET #zz: TEST YOURSELF"
9160  DATA 01,06,30,05," "
9170  DATA 01,36,20,04," "
9180  DATA 02,06,20,00,"Directions"
9190  DATA 02,24,40,00,"1. WRITE the definition of each"
9200  DATA 03,24,40,00,"symbol when you take the self test."
9210  DATA 05,24,40,00,"2. If you want a memory aid ..."
9220  DATA 06,53,11,00,"Go to :nxt0"
9230  DATA 07,24,40,00,"3. Refer to the memory aids only"
9240  DATA 08,24,40,00,"when you can't think of the definition."
9250  DATA 09,24,37,04," "
9260  DATA 10,06,20,00,"Self Test"
9270  DATA 16,36,20,00,"For Answers..."
9280  DATA 16,53,11,00,"Go to :skf1"
9290  DATA 00,00,00,09," "
9300  DATA 01,06,30,00,"SYMBOL SET #zz: TEST YOURSELF"
9310  DATA 01,06,30,05," "
9320  DATA 01,36,20,04," "
9330  DATA 02,06,19,00,"Symbols"
9340  DATA 02,26,30,00,"Memory Aids"
9350  DATA 02,06,19,05," "
9360  DATA 02,26,30,05," "
9370  DATA 16,48,16,00,"Go Back to :skb1"
9380  DATA 00,00,00,09," "
9390  DATA 01,06,30,00,"SYMBOL SET #zz: TEST YOURSELF"
9400  DATA 01,06,30,05," "
9410  DATA 01,36,20,04," "
9420  DATA 02,06,10,00,"Symbols"
9430  DATA 02,36,07,00,"Answers"
9440  DATA 02,06,19,05," "
9450  DATA 02,26,30,05," "
9460  DATA 16,53,11,00,"Go to :nxt0"
9470  DATA 00,00,00,09," "
```

6.16 ATH1Ø6.9

The 'ATH1Ø6.9' subprogram contains the format definitions for the following pages:

- Test Yourself (End of Test Directions)
- Test Yourself (Rest Directions)

The remainder of this section presents the 'ATH1Ø6.9' subprogram listing.

AUTHOR System: Phase 1 - ATH106.9 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH106.9)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 6 .9
9020  REM **
9030  REM **    ATH106.9 contains format descriptions for
9040  REM **        - Symbol Set; Test Yourself E-O-T    (7.7.5)
9050  REM **        - Symbol Set; End of Symbol Set      (7.8.0)
9060  REM **
9070  REM ********************************************************
9080  REM **
9090  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9100  GOSUB ' 050                                        : REM RETURN JUMP
9110  REM **
9120  REM **                                               FORMAT DESCRIPTIONS
9130  REM **
9140  DATA 01,06,30,00,"SYMBOL SET #zz: TEST YOURSELF"
9150  DATA 01,06,30,05," "
9160  DATA 01,36,20,04," "
9170  DATA 02,06,20,00,"End of Test"
9180  DATA 03,06,20,00,"Directions"
9190  DATA 02,24,40,00,"1. If you referred to any memory aids"
9200  DATA 03,24,40,00,"during this test, or if you missed"
9210  DATA 04,24,40,00,"one or more definitions, you need to"
9220  DATA 05,24,40,00,"continue to practice:"
9230  DATA 06,28,36,00,"-skip symbols you already know"
9240  DATA 07,28,36,00,"-start with the most difficult"
9250  DATA 08,28,36,00," symbols for you"
9260  DATA 09,28,36,00,"-do the self test after you"
9270  DATA 10,28,36,00," practice each time"
9280  DATA 11,24,36,00,"To Practice..."
9290  DATA 11,48,16,00,"Go Back to :skb3"
9300  DATA 13,24,40,00,"2. If you correctly defined all the"
9310  DATA 14,24,40,00,"symbols..."
9320  DATA 16,24,40,00,"For next set of symbols..."
9330  DATA 16,53,11,00,"Go to :nxt0"
9340  DATA 00,00,00,09," "
9350  DATA 01,06,30,00,"SYMBOL SET #zz: TEST YOURSELF"
9360  DATA 01,06,30,05," "
9370  DATA 01,36,20,04," "
9380  DATA 02,06,20,00,"Congratulations!"
9390  DATA 02,24,40,00,"You now know the definitions of the"
9400  DATA 03,24,40,00,"symbols of Set #zz."
9410  DATA 05,24,40,00,"You have completed your first study"
9420  DATA 06,24,40,00,"of this set of symbols."
9430  DATA 07,24,37,04," "
9440  DATA 08,06,20,00,"Rest"
9450  DATA 08,24,40,00,"Relax. A rest between learning tasks"
9460  DATA 09,24,40,00,"makes you a better learner."
9470  DATA 11,24,40,00,"Think of something relaxing for a"
9480  DATA 12,24,40,00,"moment or two. Then..."
9490  DATA 16,53,11,00,"Go to :nxt0"
9500  DATA 00,00,00,09," "
```

6.17 ATH1Ø7.Ø

The 'ATH1Ø7.Ø' subprogram contains the format definition for the Practice Defining All the Symbols (Overview and Direction) page.

The remainder of this section presents the 'ATH1Ø7.Ø' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH107.0)  Version > 2.0 T/VP  780317
9010  REM ** Format Description ********************************
9020  REM **
9030  REM **    ATH107.0 contains format descriptions for
9040  REM **        - Symbol Set; Practice ; Directions (8.1)
9050  REM **
9060  REM ******************************************************
9070  REM **
9080  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                      : REM RETURN JUMP
9100  REM **
9110  REM **                                             FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,30,00,"PRACTICE DEFINING ALL SYMBOLS"
9140  DATA 01,06,30,05," "
9150  DATA 01,36,20,04," "
9160  DATA 02,06,20,00,"Overview"
9170  DATA 05,06,20,00,"Directions"
9180  DATA 02,24,40,00,"Now you will practice recalling all"
9190  DATA 03,24,40,00,"## symbols in a single exercise."
9200  DATA 04,24,37,04," "
9210  DATA 05,24,40,00,"Read all the directions before"
9220  DATA 06,24,40,00,"starting the excerise."
9230  DATA 07,24,40,00,"1. Recall the definitions of each"
9240  DATA 08,24,40,00,"symbol in the exercise on frame :nxt0"
9250  DATA 09,24,40,00,"2. WRITE your first impression of"
9260  DATA 10,24,40,00,"each symbol's definition."
9270  DATA 11,24,40,00,"3. Recall the memory aid if the"
9280  DATA 12,24,40,00,"definition is difficult to remember."
9290  DATA 13,24,40,00,"Then recall the definition."
9300  DATA 14,24,40,00,"4. Check your answers only if you"
9310  DATA 15,24,40,00,"are in doubt."
9320  DATA 16,53,11,00,"Go to :nxt0"
9330  DATA 00,00,00,09," "
```

6.18 ATH1Ø7.1

The 'ATH1Ø7.1' subprogram contains the format definitions for the following pages:

- Practice Defining All the Symbols (Practice Exercises)
- Practice Defining All the Symbols (Answers)

The remainder of this section presents the 'ATH1Ø7.1' subprogram listing.

AUTHOR System: Phase 1 - ATH107.1 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH107.1)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 7 .1
9020  REM **
9030  REM **    ATH107.1 contains format descriptions for
9040  REM **         - Symbol Set; Practice ; Symbols    (8.2)
9050  REM **         - Symbol Set; Practice ; Answers    (8.3)
9060  REM **
9070  REM ******************************************************
9080  REM **
9090  GOSUB ' 205 (3)                                  : REM LOAD FIRST PAGE
9100  GOSUB ' 050                                      : REM RETURN JUMP
9110  REM **
9120  REM **                                             FORMAT DESCRIPTIONS
9130  REM **
9140  DATA 01,06,30,00,"PRACTICE DEFINING ALL SYMBOLS"
9150  DATA 01,06,30,05," "
9160  DATA 01,36,20,04," "
9170  DATA 02,06,20,00,"Practice"
9180  DATA 03,06,20,00,"Symbols"
9190  DATA 16,36,28,00,"To check Answers Go to :nxt1"
9200  DATA 00,00,00,09," "
9210  DATA 01,06,30,00,"PRACTICE DEFINING ALL SYMBOLS"
9220  DATA 01,06,30,05," "
9230  DATA 01,36,20,04," "
9240  DATA 02,06,10,00,"Symbols"
9250  DATA 02,36,07,00,"Answers"
9260  DATA 02,06,19,05," "
9270  DATA 02,26,30,05," "
9280  DATA 16,53,11,00,"Go to :nxt0"
9290  DATA 00,00,00,09," "
```

6.19 ATH1Ø7.2

The 'ATH1Ø7.2' subprogram contains the format definitions for the following pages:

- Practice Defining All the Symbols (Ending Directions)
- Practice Defining All the Symbols (Practice Directions)

The remainder of this section presents the 'ATH1Ø7.2' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH107.2)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 7 .2
9020  REM **
9030  REM **    ATH107.2 contains format descriptions for
9040  REM **        - Symbol Set; Practice ; Direct/1   (8.4)
9050  REM **        - Symbol Set; Practice ; Direct/2   (8.5)
9060  REM **
9070  REM ******************************************************
9080  REM **
9090  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9100  GOSUB ' 050                                      : REM RETURN JUMP
9110  REM **
9120  REM **                                             FORMAT DESCRIPTIONS
9130  REM **
9140  DATA 01,06,30,00,"PRACTICE DEFINING ALL SYMBOLS"
9150  DATA 01,06,30,05," "
9160  DATA 01,36,20,04," "
9170  DATA 02,06,20,00,"Directions"
9180  DATA 02,24,40,00,"1. If you gave the definitions"
9190  DATA 03,24,40,00,"of all of these symbols correctly"
9200  DATA 04,24,40,00,"without pausing..."
9210  DATA 04,53,11,00,"Go to :skf1"
9220  DATA 06,24,40,00,"2. If you paused in defining some"
9230  DATA 07,24,40,00,"of the symbols or you could not"
9240  DATA 08,24,40,00,"think of their definitions "
9250  DATA 09,24,40,00,"at all..."
9260  DATA 16,53,11,00,"Go to :nxt0"
9270  DATA 00,00,00,09," "
9280  DATA 01,06,30,00,"PRACTICE DEFINING ALL SYMBOLS"
9290  DATA 01,06,30,05," "
9300  DATA 01,36,20,04," "
9310  DATA 02,06,20,00,"Directions"
9320  DATA 02,24,40,00,"1. Keep practicing until you can"
9330  DATA 03,24,40,00,"define all the symbols without pausing."
9340  DATA 04,24,40,00,"2. Practice defining the most"
9350  DATA 05,24,40,00,"difficult symbols for you more than"
9360  DATA 06,24,40,00,"the easier symbols."
9370  DATA 07,24,40,00,"3. Change the order of going through"
9380  DATA 08,24,40,00,"the practice symbol list as you study."
9390  DATA 09,24,40,00,"Examples of how you might vary the"
9400  DATA 10,24,40,00,"order of studying the symbol codes "
9410  DATA 11,24,40,00,"in the list:"
9420  DATA 12,24,40,00,"-----> <----- -----> <-----"
9430  DATA 13,24,40,00,"<----- -----> -----> <-----"
9440  DATA 14,24,40,00,"-----> <----- -----> <-----"
9450  DATA 15,32,32,00,"To practice...  Go back to "
9460  DATA 15,59,05,00,":skb1"
9470  DATA 16,32,32,00,"After practicing...  Go to "
9480  DATA 16,59,05,00,":nxt0"
9490  DATA 00,00,00,09," "
```

6.20 ATH1Ø8.Ø

The 'ATH1Ø8.Ø' subprogram contains the format definition for the Criterion Test (Directions) page.

The remainder of this section presents the 'ATH1Ø8.Ø' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH108.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 8 .0
9020  REM **
9030  REM **    ATH108.0 contains format descriptions for
9040  REM **        - Criterion Test ;  Directions     (9.1)
9050  REM **
9060  REM ******************************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,14,00,"CRITERION TEST"
9140  DATA 01,06,14,05," "
9150  DATA 01,20,36,04," "
9160  DATA 02,06,20,00,"Ready?"
9170  DATA 02,24,40,00,"You should now be ready to take"
9180  DATA 03,24,40,00,"the criterion test."
9190  DATA 04,24,37,04," "
9200  DATA 05,06,20,00,"Directions"
9210  DATA 05,24,40,00,"1. WRITE the definition of each test"
9220  DATA 06,24,40,00,"symbol."
9230  DATA 08,24,40,00,"2. Do not check your answers until"
9240  DATA 09,24,40,00,"you have completed the entire test."
9250  DATA 11,24,35,00,"3. To be correct your answer must"
9260  DATA 12,24,38,00,"contain: (1) the concepts and (2) the"
9270  DATA 13,24,35,00,"important words in the definition."
9280  DATA 16,35,29,00,"For test symbols  Go to :nxt0"
9290  DATA 00,00,00,09," "
```

6.21 ATH1Ø8.1

The 'ATH1Ø8.1' subprogram contains the format definitions for the following pages.

- Criterion Test (Tests)
- Criterion Test (Answers)
- Criterion Test (End of Test Directions)
- Criterion Test (Congratulations)

The remainder of this section presents the 'ATH1Ø8.1' subprogram listing.

AUTHOR System: Phase 1 - ATH108.1 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH108.1)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 8 .1
9020  REM **
9030  REM **    ATH108.1 contains format descriptions for
9040  REM **        - Criterion Test ;  Symbols         (9.2)
9050  REM **        - Criterion Test ;  Answers         (9.3)
9060  REM **        - Criterion Test ;  End of Test     (9.4)
9070  REM **        - Criterion Test ; Congradulations (9.5)
9080  REM **
9090  REM ******************************************************
9100  REM **
9110  GOSUB ' 205 (1)                                  : REM LOAD FIRST PAGE
9120  GOSUB ' 050                                      : REM RETURN JUMP
9130  REM **
9140  REM **                                             FORMAT DESCRIPTIONS
9150  REM **
9160  DATA 01,06,18,00,"CRITERION TEST #zz"
9170  DATA 01,06,18,05," "
9180  DATA 01,24,32,04," "
9190  DATA 16,53,11,00,"Go to :nxt0"
9200  DATA 00,00,00,09," "
9210  DATA 01,06,18,00,"CRITERION TEST #zz"
9220  DATA 01,06,18,05," "
9230  DATA 01,24,32,04," "
9240  DATA 02,06,10,00,"Symbols"
9250  DATA 02,36,07,00,"Answers"
9260  DATA 02,06,19,05," "
9270  DATA 02,26,30,05," "
9280  DATA 16,53,11,00,"Go to :nxt0"
9290  DATA 00,00,00,09," "
9300  DATA 01,06,18,00,"CRITERION TEST #zz"
9310  DATA 01,06,18,05," "
9320  DATA 01,24,32,04," "
9330  DATA 02,06,20,00,"End of Test"
9340  DATA 03,06,20,00,"Directions"
9350  DATA 02,22,42,00,"* If you correctly defined all of the"
9360  DATA 03,24,40,00,"symbols..."
9370  DATA 03,53,11,00,"Go to :end*"
9380  DATA 04,22,42,00,"* If you missed one or more answers:"
9390  DATA 05,30,34,00,"1.Make a list of symbols"
9400  DATA 06,30,34,00,"that you missed."
9410  DATA 07,30,34,00,"2.Practice defining these symbols."
9420  DATA 08,30,34,00,"3.Start with the most difficult"
9430  DATA 09,30,34,00,"symbols for you."
9440  DATA 10,30,34,00,"4.Continue to practice until you"
9450  DATA 11,30,34,00,"can correctly define them."
9460  DATA 12,30,34,00,"5.Take the next criterion test..."
9470  DATA 13,30,34,00,"2.Go back to practicing"
9480  DATA 14,30,34,00,"all the symbols..."
9490  DATA 15,53,11,00,"Go to :prac"
9500  DATA 16,53,11,00,"Go to :nxt0"
9510  DATA 00,00,00,09," "
```

```
9520  DATA 01,06,16,00,"CONGRATULATIONS!"
9530  DATA 01,06,16,05," "
9540  DATA 01,22,34,04," "
9550  DATA 03,24,40,00,"You have shown that you know all"
9560  DATA 04,24,40,00,"of the symbols in this program."
9570  DATA 06,24,40,00,"Knowing these symbols will make "
9580  DATA 07,24,40,00,"you a better "
9590  DATA 16,53,11,00,"Go to :nxt0"
9600  DATA 00,00,00,09," "
```

6.22 ATH1Ø9.Ø

The 'ATH1Ø9.Ø' subprogram contains the format definition for the Suggestions for Refresher Training page.

The remainder of this section presents the 'ATH1Ø9.Ø' subprogram listing.

```
9000  %## Data Subprogram > AUTHOR/1 (ATH109.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 0 9 .0
9020  REM **
9030  REM **    ATH109.0 contains format descriptions for
9040  REM **        - Suggestions for Refresher Train  (10.0)
9050  REM **
9060  REM ********************************************************
9070  REM **
9080  GOSUB ' 205 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,34,00,"SUGGESTIONS FOR REFRESHER TRAINING"
9140  DATA 01,06,34,05," "
9150  DATA 01,40,16,04," "
9160  DATA 02,06,20,00,"People"
9170  DATA 03,06,20,00,"Forget"
9180  DATA 07,06,20,00,"Suggestions"
9190  DATA 02,24,40,00,"You will need to refresh your"
9200  DATA 03,24,40,00,"memory once in a while if you are"
9210  DATA 04,24,40,00,"to recognize these symbols"
9220  DATA 05,24,40,00,"on-the-job."
9230  DATA 06,24,37,04," "
9240  DATA 07,24,40,00,"1. Review the symbols before you"
9250  DATA 08,24,40,00,"go to bed TONIGHT."
9260  DATA 10,24,40,00,"2. Review them again in ONE WEEK."
9270  DATA 12,24,40,00,"3. If you seldom use these symbols"
9280  DATA 13,24,40,00,"on-the-job, review them EVERY MONTH."
9290  DATA 16,53,11,00,"Go to :nxt0"
9300  DATA 00,00,00,09," "
9310  DATA 01,06,34,00,"SUGGESTIONS FOR REFRESHER TRAINING"
9320  DATA 01,06,34,05," "
9330  DATA 01,40,16,04," "
9340  DATA 02,06,20,00,"Suggestions"
9350  DATA 02,24,40,00,"4. Refresh your memory while you"
9360  DATA 03,24,40,00,"still remember. Don't wait until "
9370  DATA 04,24,40,00,"you forget. This will be easier than"
9380  DATA 05,24,40,00,"learning these symbols again after"
9390  DATA 06,24,40,00,"you have forgotten them."
9400  DATA 08,24,40,00,"5. Check for accuracy during refresher"
9410  DATA 09,24,40,00,"exercises. Your memory plays tricks"
9420  DATA 10,24,40,00,"on you. Use the criterion test first,"
9430  DATA 11,24,40,00,"if it is available."
9440  DATA 13,24,40,00,"6. Refresh your memory of the"
9450  DATA 14,24,40,00,"definitions of these symbols more"
9460  DATA 15,24,40,00,"often than is needed to just get by."
9470  DATA 16,44,20,00,"END OF PROGRAM"
9480  DATA 00,00,00,09," "
9490  DATA 01,25,14,00,"END OF FILE"
9500  DATA 08,01,14,00,"END OF FILE"
9510  DATA 08,25,14,00,"END OF FILE"
9520  DATA 08,50,14,00,"END OF FILE"
9530  DATA 09,01,14,00,"END OF FILE"
9540  DATA 09,25,14,00,"END OF FILE"
9550  DATA 09,50,14,00,"END OF FILE"
9560  DATA 16,01,14,00,"END OF FILE"
9570  DATA 16,50,14,00,"END OF FILE"
9580  DATA 00,00,00,09," "
```

6.23 ATH11Ø.Ø

The 'ATH11Ø.Ø' subprogram contains the format definitions for the Index page.

The remainder of this section presents the 'ATH11Ø.Ø' subprogram listing.

AUTHOR System: Phase 1 - ATH110.0 - Format

```
9000  %## Data Subprogram > AUTHOR/1 (ATH110.0)  Version > 2.0 T/VP  780317
9010  REM **  Data Subroutine  ************************************** 1 0 .0
9020  REM **
9030  REM **    ATH110.0 contains format descriptions for
9040  REM **        - Index of Learning Program          (11.0)
9050  REM **
9060  REM ************************************************************
9070  REM **
9080  GOSUB ' 209 (1)                                    : REM LOAD FIRST PAGE
9090  GOSUB ' 050                                        : REM RETURN JUMP
9100  REM **
9110  REM **                                               FORMAT DESCRIPTIONS
9120  REM **
9130  DATA 01,06,05,00,"INDEX"
9140  DATA 01,06,05,05," "
9150  DATA 01,11,45,04," "
9160  DATA 02,49,12,00,"Page Address"
9170  DATA 02,49,12,05," "
9180  DATA 03,06,40,00,"Introduction"
9190  DATA 04,06,54,04," "
9200  DATA 05,06,40,00,"Optional Criterion Test"
9210  DATA 06,06,54,04," "
9220  DATA 07,06,40,00,"Organization of Learning Program"
9230  DATA 08,06,54,04," "
9240  DATA 09,06,40,00,"Symbol Sets:"
9250  DATA 00,00,00,09," "
```

7.Ø FORMAT LISTINGS

The format listings presented in this section are the formats used in creating the learning module. These formats are defined by the data subprograms presented in section 6.Ø of this report.

AUTHOR: An Automatic Authoring System Version: 2.0

INTRODUCTION

Learning

Objective

This lesson will help you learn easily and quickly.

FOR BEST RESULTS, FOLLOW THE DIRECTIONS.

Index

When you need an index to locate practice exercises for refresher training... Go to :indx

To continue Go to :nxt0

AUTHOR: An Automatic Authoring System Version: 2.0

IMPORTANCE OF THIS PROGRAM TO YOU

Importance

of Symbols

How Skill

Will be Used

This program will help you to:

Go to :nxt0

EXAMPLE OF HOW THE SYMBOLS ARE USED

Go to :nxt0

AUTHOR: An Automatic Authoring System Version: 2.0

SYMBOL PRETEST

Purpose — The pretest will help you decide whether you should try the criterion test at this time.

Directions — 1. You may already know some or all of these

2. Look through the list of symbols below and see how many you can define.

Symbols

Go to :nxt0

OPTIONAL CRITERION TEST

Choose Item #1 1. If you think that there are THREE
Or Item #2 OR MORE of these symbols that YOU DO NOT KNOW, don't waste time taking the criterion test. Go directly to the learning program... Go to :skf2

2. If you think you know all the symbols; or you missed only 1 or 2 take the criterion test and see if you can pass it...

Go to :nxt0

OPTIONAL CRITERION TEST

Directions

1. Recall the definition of each symbol on the next frame(s).

2. Write each definition on a piece of paper.

3. You must recall the official definition of the symbol to be correct (as found in

Go to :nxt0

AUTHOR: An Automatic Authoring System Version: 2.0

OPTIONAL CRITERION TEST: SYMBOLS

Go to :nxt0

OPTIONAL CRITERION TEST

Directions 1. Check your answers now.

2. Put an X through your wrong answers.

Symbols Answers

Go to :nxt0

OPTIONAL_CRITERION_TEST

Did you pass?

1. Did you miss ## or less of the symbol definitions? If so, then learn the ones that you missed by studying them on :skb1 ...

Then Go to :clos

2. If you missed more than ## of the symbol definitions, complete this learning program - Now ...

Go to :nxt0

ORGANIZATION OF THE LEARNING PROGRAM

Overview The list of symbols has been divided into ## practice sets. You will learn each set separately, using the 'Steps for Easy Learning.' Then you will practice all ## symbols in the same exercise.

Steps for Easy Learning

:dia0

!Note the Differences! Among the Symbols	!Learn the symbol Definitions	!Practice! Defining the Symbols	!Test Yourself	Did You Pass ?

No

Go to :nxt0

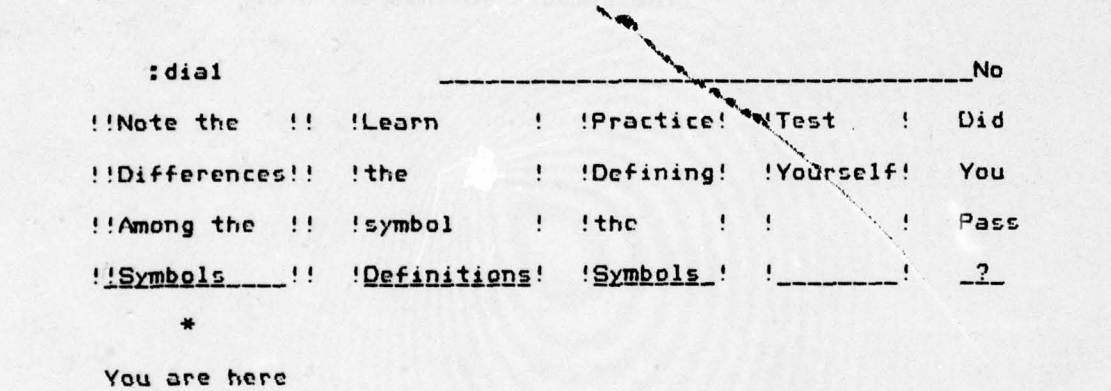
SYMBOL SET #zz

The symbols in this set are:

You will learn to recognize and define these symbols in the next few frames.

Go to :nxt0

SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS



Go to :nxt0

SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS

Introduction Each symbol represents a different thing, yet some of the symbols look very much alike. The symbols presented below may be confused with each other:

Carefully note what makes the symbols above different.
At this point in the learning program, look only at the differences among the symbols. Later on in the program you will be given definitions associated with each symbol.

Go to :nxt0

SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS

Purpose

This exercise will help you to note the differences among those symbols that are often confused. After you finish with this exercise, you should not confuse similar symbols.

Quiz Directions

1. Read all directions before going to the next frame.
2. Look carefully at this symbol:

3. Write the item number of each example of on paper.
4. Do not look back at this frame until you have finished the quiz for this symbol.

Go to :nxt2

SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS

Quiz

Symbols

Go to :nxt0

SYMBOL SET #zz: NOTE DIFFERENCES AMONG SYMBOLS

Quiz Symbol Examples

Answers

What Now?

If you frequently missed examples of these symbols, repeat this exercise . . . Go to :skb1

If you identified most all of the symbols . . .

Go to :nxt0

AUTHOR: An Automatic Authoring System Version: 2.0



SYMBOL SET #zz: LEARN SYMBOL DEFINITIONS

:dia2

Note the Differences Among the Symbols	Learn the symbol Definitions	Practice Defining the Symbols	Test Yourself	Did You Pass ?
	*			No

You are here

Go to :nxt0



SYMBOL SET #zz: LEARN SYMBOL DEFINITIONS

Directions

1. Look carefully on the next frame at each symbol and its definition.

2. Repeat the definition to yourself as you look at the symbol.

3. Study the memory aid to help you remember the definition.

4. Study each symbol and its definition 2 or 3 times.

Go to :nxt0

SYMBOL SET #zz: LEARN SYMBOL DEFINITIONS

Definition	Symbol	Memory Aid

Go to :nxt0

AUTHOR: An Automatic Authoring System Version: 2.0

SYMBOL SET #zz: REST

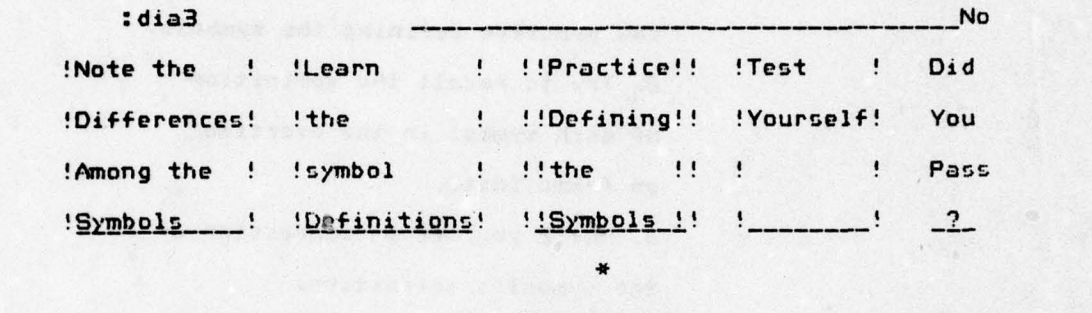
Directions -- Relax for a moment or two.
Get your mind off your study.

-- Think of something relaxing.

-- You will learn more quickly
when you go back to your study
if you rest.

Go to :nxt0

SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS



Go to :nxt0

SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS

Directions

1. Read all directions before you practice defining the symbols.
2. Try to recall the definition of each symbol in the exercise on frame :nxt0.
3. WRITE your first impression of the symbol's definition.
4. If the definition is difficult to remember, recall the memory aid first, then recall the definition.
5. Check your answer immediately in the answer section below the symbols.

Go to :nxt0

SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS

Practice

Symbols

Symbol	Definition

Go to :nxt0

SYMBOL SET #zz: PRACTICE DEFINING THE SYMBOLS

Directions

1. Keep practicing until you feel you know the definition of each symbol.
2. Practice defining the most difficult symbols for you more than the easier symbols.
3. Change the order of going through the practice symbol list as you study. Examples of how you might vary the order of studying the symbol codes in the list:

-----> <----- -----> <-----

<----- -----> -----> <-----

-----> <----- -----> <-----

To practice... Go back to :skb1

After practicing... Go to :nxt0

SYMBOL SET #zz: WANT TO REST?

Yes — If you are tired now, get your mind off studying symbols.

Relax for a minute!

After your minute's rest...

Go to :nxt0

No — If you do not want to rest now...

Go to :nxt0



SYMBOL SET #zz: TEST YOURSELF

:dia4

!Note the Differences! Among the Symbols	!Learn the symbol Definitions	!Practice! !Defining! !the Symbols	!!Test !!Yourself!! !! !!	Did You Pass ?

No

*

You are here

Go to :nxt0



AD-A061 611 TRAINING ANALYSIS AND EVALUATION GROUP (NAVY) ORLANDO FLA F/G 5/2
COMPUTER-AIDED AUTHORING SYSTEM (AUTHOR) USER'S GUIDE. VOLUME I--ETC(U)
JUN 78 C R GUITARD

UNCLASSIFIED TAEG-59-VOL-2 NL

10 OF 10
ADA
061811



END
DATE
FILMED
1 -79
DDC

SYMBOL SET #zz: TEST YOURSELF

Directions

1. WRITE the definition of each symbol when you take the self test.

2. If you want a memory aid ...

Go to :nxt0

3. Refer to the memory aids only when you can't think of the definition.

Self Test

For Answers... Go to :skf1

AUTHOR: An Automatic Authoring System Version: 2.0

SYMBOL SET #zz: TEST YOURSELF

Symbols Memory Aids

Go Back to :skb1

SYMBOL SET #zz: TEST YOURSELF

Symbols Answers

Go to :nxt0

SYMBOL SET #zz: TEST YOURSELF

End of Test Directions

1. If you referred to any memory aids during this test, or if you missed one or more definitions, you need to continue to practice:

   -skip symbols you already know

   -start with the most difficult symbols for you

   -do the self test after you practice each time

To Practice... Go Back to :skb3

2. If you correctly defined all the symbols...

For next set of symbols... Go to :nxt0

SYMBOL SET #zz: TEST YOURSELF

Congratulations! You now know the definitions of the symbols of Set #zz.

You have completed your first study of this set of symbols.

Rest Relax. A rest between learning tasks makes you a better learner.

Think of something relaxing for a moment or two. Then...

Go to :nxt0

PRACTICE DEFINING ALL SYMBOLS

Overview — Now you will practice recalling all ## symbols in a single exercise.

Directions — Read all the directions before starting the excerise.

1. Recall the definitions of each symbol in the exercise on frame :nxt0
2. WRITE your first impression of each symbol's definition.
3. Recall the memory aid if the definition is difficult to remember. Then recall the definition.
4. Check your answers only if you are in doubt.

Go to :nxt0

PRACTICE DEFINING ALL SYMBOLS

Practice

Symbols

To check Answers Go to :nxt1

PRACTICE DEFINING ALL SYMBOLS

Symbols Answers

Go to :nxt0

PRACTICE DEFINING ALL SYMBOLS

Directions

1. If you gave the definitions of all of these symbols correctly without pausing... Go to :skf1

2. If you paused in defining some of the symbols or you could not think of their definitions at all...

Go to :nxt0

PRACTICE DEFINING ALL SYMBOLS

Directions

1. Keep practicing until you can define all the symbols without pausing.
2. Practice defining the most difficult symbols for you more than the easier symbols.
3. Change the order of going through the practice symbol list as you study.

Examples of how you might vary the order of studying the symbol codes in the list:

-----> <----- -----> <-----

<----- -----> -----> <-----

-----> <----- -----> <-----

To practice... Go back to :skb1

After practicing... Go to :nxt0

CRITERION TEST

Ready? You should now be ready to take the criterion test.

Directions 1. WRITE the definition of each test symbol.

2. Do not check your answers until you have completed the entire test.

3. To be correct your answer must contain: (1) the concepts and (2) the important words in the definition.

For test symbols Go to :nxt0

AUTHOR: An Automatic Authoring System Version: 2.0

CRITERION TEST #zz

Go to :nxt0

CRITERION TEST #zz

Symbols Answers

Go to :nxt0

CRITERION TEST #zz

End of Test Directions

* If you correctly defined all of the symbols... Go to :end*

* If you missed one or more answers:

1.Make a list of symbols that you missed.

2.Practice defining these symbols.

3.Start with the most difficult symbols for you.

4.Continue to practice until you can correctly define them.

5.Take the next criterion test...

2.Go back to practicing all the symbols...

Go to :prac

Go to :nxt0

CONGRATULATIONS!

You have shown that you know all
of the symbols in this program.

Knowing these symbols will make
you a better

Go to :nxt0

SUGGESTIONS FOR REFRESHER TRAINING

People Forget

You will need to refresh your memory once in a while if you are to recognize these symbols on-the-job.

Suggestions

1. Review the symbols before you go to bed TONIGHT.

2. Review them again in ONE WEEK.

3. If you seldom use these symbols on-the-job, review them EVERY MONTH.

Go to :nxt0

SUGGESTIONS FOR REFRESHER TRAINING

Suggestions

4. Refresh your memory while you still remember. Don't wait until you forget. This will be easier than learning these symbols again after you have forgotten them.

5. Check for accuracy during refresher exercises. Your memory plays tricks on you. Use the criterion test first, if it is available.

6. Refresh your memory of the definitions of these symbols more often than is needed to just get by.

END OF PROGRAM

END OF FILE

END OF FILE END OF FILE END OF FILE

END OF FILE END OF FILE END OF FILE

END OF FILE END OF FILE

INDEX

Page Address

Introduction

Optional Criterion Test

Organization of Learning Program

Symbol Sets:

8.Ø COMPRESSING PROGRAM SOURCE CODE

The AUTHOR System program contains a large amount of internal program documentation. Because of this documentation the programs must first be compressed (removal of unnecessary blanks and all comments) before the programs may be used.

After compressing the AUTHOR System programs, a few of the programs must first be combined and saved as one. A list of these programs is given in figure 8.1.

The compression of the programs is done by using the compression utility provided by WANG. For instructions on how to use the ISS compression utility see the WANG Integrated Support System User Manual, Chapter 11. TAEG will provide a tape of the compressed program to DOD activities upon request.

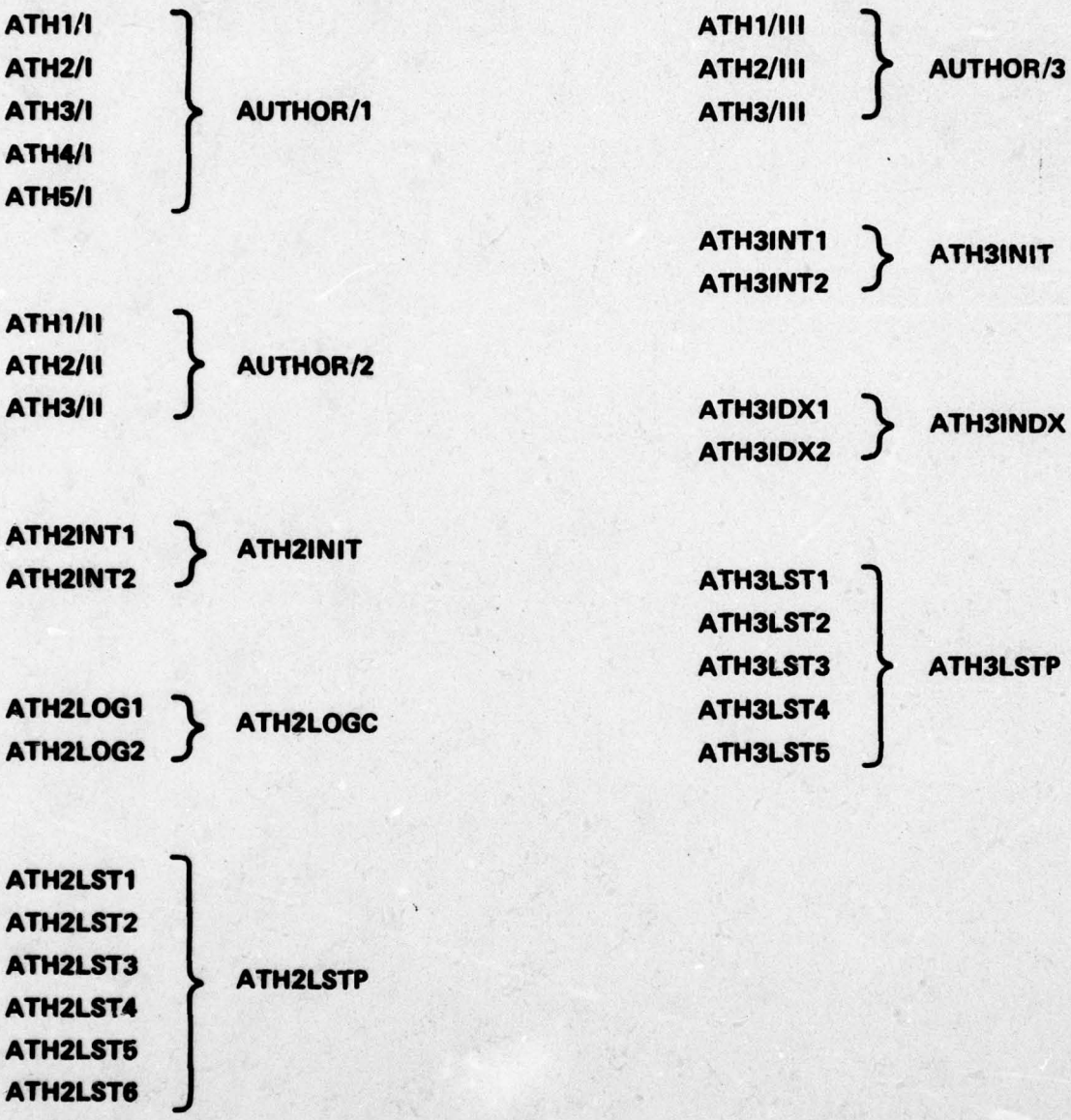


FIGURE 8.1. COMBINATION OF MULTIPLE PROGRAM FILES

TAEG Report No. 59

Volume II

DISTRIBUTION LIST

COMDTNSRDC (Sam Rainey) (5 copies)
CO NPRDC (Dr. W. Wulfeck II)
CO NAVEDTRASUPPCENPAC (Harry Stevenson)
OIC Naval Instructional Program Development Detachment (Bob King)